XII Escola Regional de Computação Bahia Alagoas Sergipe

De 24 a 27 de abril de 2012, UNIVASF Juazeiro - BA

www.erbase2012.univasf.edu.br
ISSN 2177-4692

Comissão Organizadora ERBASE 2012

Jorge Luis Cavalcanti Ramos
UNIVASF– Coordenador Geral

Alexandre Roberto de Souza Correia
IF Sertão – PE

Dinani Gomes Amorim
UNEB/FACAPE

Eudis Oliveira Teixeira
IF Sertão – PE

Luis Nícolas de Amorim Trigo
IF Sertão – PE

Maria Celimar da Silva
FACAPE

Mario Godoy Neto
UNIVASF

Ricardo José Rocha Amorim
UNEB/FACAPE

Rômulo Calado Pantaleão Câmara
UNIVASF

Rossana de Paula Junqueira Almeida
FACAPE / IF Sertão – PE

Vânia Cristina Lasalvia
FACAPE

www.erbase2012.univasf.edu.br
ISSN 2177-4692

Editor e Organizador dos Anais

Mario Godoy Neto
UNIVASF

Monitores Voluntários

Cleidson Alves
UNIVASF

Eliezer Andrade
UNIVASF

Leonardo Campello
UNIVASF

www.erbase2012.univasf.edu.br

ISSN 2177-4692

A ERBASE 2012 - Computação nas águas do Sertão

As Escolas Regionais de Computação são eventos promovidos pela Sociedade Brasileira de Computação (SBC) e executados no âmbito das Secretarias Regionais da SBC. No Brasil existem vinte e uma secretarias regionais, das quais sete são localizadas na região Nordeste. Estas Escolas visam divulgar localmente novas ideias, debater e promover a atualização profissional nos domínios da computação para alunos de graduação e pós-graduação, professores e profissionais da área.

A ERBASE é um fórum para pesquisadores, professores, alunos, desenvolvedores e usuários, de Computação e áreas afins onde os interessados no tema terão oportunidade de trocar ideias e experiências, realizar debates e fomentar a ciência na área. É um evento multi-institucional que conta com a parceria das principais instituições de ensino superior dos estados da Bahia, Alagoas e Sergipe, tendo como público-alvo alunos de graduação e pós-graduação em computação e áreas afins.

A ERBASE é um evento anual já consolidado, que ocorre desde 2001 e provê várias atividades de interesse, como palestras, mini-cursos, laboratórios e painéis nas diversas áreas da tecnologia da informação. No ano de 2012, será realizada a décima segunda edição da ERBASE, sendo esta organizada pelas instituições Universidade Federal do Vale do São Francisco – UNIVASF, Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano – IF-Sertão Pernambucano e pela Faculdade de Ciências Aplicadas e Sociais de Petrolina – FACAPE.

www.erbase2012.univasf.edu.br

ISSN 2177-4692

Agradecimentos

Aos palestrantes.
O conhecimento torna-se valioso e indispensável quando é compartilhado. A difusão do saber, a troca de ideias e experiências é grande legado da era da informação. Cada palestrante trouxe muito mais que sua apresentação. Trouxe novos horizontes, desafios e possibilidades para os profissionais de tecnologia da informação de toda região.

Aos autores.
São os protagonistas do evento, seus trabalhos foram avaliados e validados por uma criteriosa comissão de programa e, sendo aprovados, indica que vocês, jovens ou experientes pesquisadores, estão contribuindo fortemente com o desenvolvimento da ciência.

Aos inscritos.
Representam a razão da existência do evento. Não há show sem plateia, nem apresentação sem ouvintes. Todo o esforço da Comissão Organizadora foi para que cada um de vocês pudesse ter a satisfação de ter investido e obtido o máximo de proveito do evento.

As instituições realizadoras.
Graças a união de três instituições: Universidade Federal do Vale do São Francisco - UNIVASF, Faculdade de Ciências Aplicadas e Sociais de Petrolina - FACAPE e o Instituto Federal do Sertão Pernambucano - IF-SERTÃO-PE, formam os pilares desta edição, oferecendo total e irrestrita dedicação a realização da ERBASE 2012.

Ao apoio e promoção.
Um evento com a chancela da Sociedade Brasileira de Computação - SBC pode ser considerado antecipadamente um evento de sucesso, pois detêm grande credibilidade junto à comunidade científica acadêmica. O alto nível dos palestrantes e toda a divulgação do evento só foi possível pelo grande apoio dado pela FAPESB, tornando o evento uma referência regional em computação.

A comissão organizadora agradece pela participação e contribuição de todos.

www.erbase2012.univasf.edu.br

ISSN 2177-4692

Sumário

Laboratórios

Capítulo

1

Desenvolvimento de Jogos Eletrônicos para HTML5

Roberto Tenorio Figueiredo, Carla Brandão de Carvalho Figueiredo

Abstract

HTML5 is a new technology that came to standardize the creation of web applications in order to adapt these systems to newer and different types of hardware on the market today. Most users of such hardware has shown interest in the use of electronic games. According to Abragames (2012), Brazil moves about 87.5 million dollars in gaming by year and those numbers only grow. With the volume of resources and the enormous revenue companies, the market increasingly demands professionals with high degree of qualification. With the shortage of health personnel in the market, training people interested in it is of extreme importance and the workshop is to help potential developers to take the first step in this market quite competitive, but also very comp.

Resumo

O HTML5 é uma nova tecnologia que veio para padronizar a criação de aplicações WEB de maneira a adaptar esses sistemas aos mais novos e diferentes tipos de hardwares existentes no mercado de hoje. Grande parte dos usuários desses hardwares tem se mostrado interessado na utilização de jogos eletrônicos. Segundo a ABRAGAMES (2012), o Brasil movimenta cerca de 87,5 milhões de reais em jogos eletrônicos por ano e esses números só crescem. Com o volume de recursos e o enorme faturamento das empresas, o mercado exige cada vez mais profissionais com alto grau de qualificação. Com a escassez desse tipo de profissional no mercado, o treinamento de pessoas interessadas nele é de estrema importância e a oficina vem para ajudar os potenciais desenvolvedores a dar o primeiro passo nesse mercado bastante competitivo, mas também muito compensativo.

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

1.1. Introdução

Esta oficina trata do desenvolvimento de jogos eletrônicos para que possa ser executado em qualquer plataforma que reconheça a tecnologia do HTML5. O processo de criação de jogos eletrônicos, feito por programadores e empresas não especializadas, costuma ser bastante desorganizado e complicado. A falta de definição exata de vários aspectos inerentes ao desenvolvimento de um software, como os requisitos a se satisfazer e as tecnologias disponíveis, faz com que o desenvolvimento de jogos siga caminhos nem sempre definidos levando muitas equipes a pular etapas fundamentais e seguir processos de acordo com o andamento do projeto.

1.2. Justificativa

O desenvolvimento de jogos eletrônicos no Brasil movimenta milhões de reais. É um mercado bastante aquecido e lucrativo, porém as empresas especializadas sentem falta de bons profissionais qualificados nas novas tecnologias. Com o surgimento do HTML5, padrão compatível com as novidades do mercado, o estudante que pretende seguir pela área de desenvolvimento de jogos, pode iniciar seu caminho aprendendo a trabalhar nessa nova ferramenta e testar seus produtos nos mais diferentes dispositivos do mercado.

1.3. Objetivos

O objetivo geral dessa da oficina é apresentar aos participantes uma IDE de programação amigável que visa facilitar a programação de softwares, principalmente jogos, para o novo padrão web, HTML5.

Como objetivos específicos têm-se:

- Apresentação do HTML5;

- Apresentação do GameMakerHTML5;

- Desenvolvimento de um jogo de nave desde sua concepção, até sua publicação para ser executado em plataformas que reconheça o padrão HTML5;

1.4. Metodologia

Este trabalho está dividido em três partes.

A primeira parte trata do desenvolvimento de jogos, desde sua idealização e formação da equipe de desenvolvimento, até sua execução pelos usuários finais, descrevendo com detalhes todas as etapas que devem ser vencidas durante a execução do projeto de um jogo eletrônico.

A segunda parte trata do HTML5. A apresentação da tecnologia, sua importância e todas as novidades que esta versão traz em relação as suas versões de outrora.

A terceira parte é o desenvolvimento de um estudo de caso, que será trabalhado na oficina. O desenvolvimento completo de um jogo de nave, seus recursos e o modelo de sua programação.

Na oficina, serão apresentadas as tecnologias envolvidas no processo de criação e o estudo de caso, onde será demonstrado o desenvolvimento de um jogo em toda sua plenitude.

1.5. Desenvolvimento de Jogos

O desenvolvimento de jogos eletrônicos é o processo de criação de um produto de entretenimento digital, no qual o usuário participará de simulações e/ou competições que estimularão seus sentidos e o imergirá em uma realidade virtual competitiva e estimulante.

O processo de criação de um jogo não é um projeto para jogadores, é um processo multidisciplinar que engloba conhecimentos de profissionais das mais diversas áreas como: computação gráfica, programação, multimídia, redes, inteligência artificial, cálculo, física e muitos outros conceitos importantes. Certamente, para a concepção de um grande jogo é necessária boa uma equipe, onde alguns membros possuem conhecimento avançado em algumas dessas ou outras áreas fundamentais.

Hoje em dia, a indústria de jogos possui equipes de desenvolvimento, com membros especializados. Entre eles, é importante destacar:

- **Programadores:** encarregados de desenvolver a parte lógica (algoritmos) do jogo;
- **Artistas:** responsáveis pelo layout do jogo. Criam objetos, personagens, texturas, ilustrações, animações, etc.
- **Projetistas de Níveis/Fases:** encarregados em elaborar as fases dos jogos, estruturando desafios e surpresas;
- **Projetistas de Jogos:** também conhecidos como Game Designers. Envolvem-se em quase todas as etapas na construção de jogos. Responsáveis pela criação das idéias para a concepção do jogo e também pela documentação de todo o jogo (com características, especificações e instruções de uso). São eles que mantém todos os demais membros da equipe em harmonia (sintonia), voltados para o cumprimento correto do projeto.
- **Planejador de Software:** Sua tarefa é dividir o projeto do jogo (elaborado pelo Game Design) em um conjunto de requisitos técnicos e estimar o tempo, custo e esforço necessário para implementar tais características.
- **Arquiteto chefe:** Trabalha em conjunto com o Planejador de Software para produzir um conjunto de especificações de módulos. É responsável pela arquitetura geral do projeto.
- **Gerente de Projeto:** Sua tarefa é balancear a carga de trabalho gerada pelo Planejador de Software e pelo Arquiteto Chefe, produzindo um planejamento eficiente e organizado. Responsável por criar o cronograma de desenvolvimento e mantê-lo em dia, monitorando, adaptando e cobrando os resultados do restante da equipe. Não deve estar envolvido diretamente com a programação e criação de arte do projeto.
- **Músicos e Sonoplastas:** Geralmente vindos de áreas relacionadas a arte e a música, são responsáveis por compor trilhas sonoras, vozes e efeitos dos jogos.

- **Testadores:** Esses profissionais entram no projeto já em suas fases finais com a responsabilidade de encontrar possíveis erros e bugs no jogo. Jamais devem fazer parte de qualquer outra fase no desenvolvimento do projeto e não devem ter muitas informações sobre o mesmo.

Depois de formada a equipe, o próximo passo do desenvolvimento é o planejamento do jogo. Esse é um passo fundamental para o sucesso do jogo e que é importante não negligenciar. Ao contrário da maioria dos softwares comerciais, a criação de um jogo não deve ser feita através de métodos cíclicos, mas sim, através de métodos de desenvolvimento ágil para pequenas equipes e processo de software pessoal para grandes equipes. Esses processos incluem, de forma geral, as seguintes etapas:

- **Pré-Produção**. Esta é a etapa da idealização do jogo e de seu conceito. Seu objetivo é produzir um documento que servirá como guia durante todo o processo de criação do jogo. É subdividida em:a

  - **Idéias:** Reuniões para exposição das idéias. Nesta fase algumas questões devem ser respondidas como: "Qual o objetivo do jogo?", "O que o jogador terá de fazer?", "Como ele vai fazer?", "O que tornará o jogo divertido?", etc.

  - **Roteirização:** É importante que seja criado o roteiro do jogo, ou seja, sua estória e que a equipe não fuga da idéia original. Novas idéias, durante as fases seguintes, poderão deixam o jogo grande demais e quase impossível de se terminar.

  - **Rascunhos do Jogo:** É a criação de desenhos preliminares (no papel ou em aplicativos editores de imagens) que mostre como serão algumas fases, personagens e itens com pequenas descrições de cada um.

  - **Detalhamento do Jogo:** O jogo precisa ser detalhado por completo. O que não for descrito, não será desenvolvido. Nesta fase, todas as perguntas devem ser esclarecidas. Ao final dessa fase, será gerado o documento único do jogo, conhecido pelas grandes indústrias como ***Design Document.***

  O *Design Document* ou Documento de Projeto do Jogo, também conhecido como DD, é como o script de um filme que informa todos os detalhes do jogo. De forma geral, o DD deve conter diversos tópicos, cada um deles com conteúdo específico. São eles:

  - Conceito: nome do jogo; apresentação resumida do jogo; público-alvo; mercado e publicidade; estilo do jogo (tipo); elementos do jogo; história do jogo; principais regras do jogo;

- Especificações Técnicas: hardware; sistema operacional; hardware mínimo; requisitos de software; gráficos;

- Especificações do Jogo: número de fases; níveis de dificuldade; vidas; descrição dos tipos ou modos de jogo; sistema de pontuação; sistema de ranking (high scores); opções de configuração; número de jogadores; recurso de carga e gravação (load e save); sistema de câmera; personagens; itens de jogo; itens de cenário; tabela de itens; evolução de fases; tabela de mensagens;

- Dispositivos de Entrada: suporte para mouse; dispositivos de entrada para os menus; dispositivos de entrada de jogo; definição de teclas e botões;

- Design Gráfico e Arte: abertura; descrição de layout de menus e telas; descrição de layout do jogo na fase; definição de fases; definição do final do jogo;

- Sonorização: definição de música nos menus; definição de música nas fases; definição dos efeitos sonoros de menus e outros; definição dos efeitos sonoros de jogo (nas fases);

- Desenvolvimento: tempo de desenvolvimento; alocação de pessoal; metas e cronograma de atividades; linguagem de programação;

- Diagramas: de fases; de estados; de progressão da jogatina;

- **Produção**. A etapa de produção é a fase onde os desenvolvedores e irão produzir os *resources* (recursos de imagens, vídeos e sons) a serem utilizados e o código-fonte do jogo. É o momento de transformar os dados do DD, em um produto jogável propriamente dito.

- **Pós-produção.** Essa última etapa é associada às entregas do produto, suas versões intermediárias, de apresentação e a versão final jogo. Entre essas versões, podemos destacar:

  - *First Playable* - É a primeira versão jogável sem grandes detalhamentos, geralmente apresentada internamente somente a equipe envolvida diretamente ou indiretamente na implementação e/ou comercialização do jogo;

  - Alfa - Fase onde todas as funcionalidades de jogabilidade estão disponíveis, mas apenas a equipe desenvolvedora tem acesso ao jogo.

  - Beta – Esta é a fase onde é entregue uma parte do jogo, já contendo todas as funcionalidades da versão final. Essa versão é acessível aos usuários

que vão, muitas vezes, dar *feedback* daquele jogo à equipe de desenvolvimento.

- *Freeze* – Esta é uma fase apenas de correção de erros. O jogo já se encontra acabado.

- *Release* – Fase onde o jogo segue para a avaliação do fabricante de console, para verificações de compatibilidade e velocidade. Ajustes poderão ser requeridos;

- *Crunch Time* – É o momento que se dá entre a aprovação final do jogo e o seu lançamento no mercado. É uma fase de divulgação de propaganda na mídia;

Durante a fase inicial de idealização e aprovação do projeto do jogo, alguns itens relevantes devem ser levados em consideração pela equipe e principalmente pelo *Game Designer*. Entre esses itens, é importante destacar: as regras para um bom jogo e o que o público alvo quer e espera desse produto.

A idéia de um bom jogo pode variar de pessoa, mas existem algumas regras que todos os bons jogos devem ter. Essas regras não compõem um manual propriamente dito, elas se originam das experiências de várias equipes de desenvolvimento e incluem boas práticas e critérios que não devem ser esquecidos. Entre eles, podemos citar:

- **Começar com uma boa idéia e uma boa história:** a história do jogo precisa ser coerente e lógica. A idéia deve ser apresentada a alguns possíveis jogadores para que o *Game Designer* veja s impressões dos mesmos sobre sua idéia, porém uma rejeição inicial não deve ser motivo de desistência, pois alguns dos mais famosos jogos eletrônicos já foram inicialmente rejeitados, como é o caso de "*The Sims*";

- **Escrever o design no papel:** a importância de se descrever todo o jogo no papel é maior do que imagina a maioria dos programadores amadores, mas levado à importância extrema por profissionais, a exemplo do jogo "Zelda, The Wind Waker", para Game Cube, que passou três anos sendo planejado e apenas oito meses para ser programado;

- **Começar pequeno:** o *Game Designer* não deve tentar direcionar a criação de seu jogo para uma plataforma desconhecida da equipe, pois não há como prever o tempo em que a equipe se sentirá confortável para assumir novas ferramentas de desenvolvimento, podendo deixar o projeto com o tempo muito maior do que o esperado;

- **Cuidar do público:** conhecer o seu público alvo e ter informações sobre ele é fundamental para se criar um jogo voltado para este público. O *Game Designer* não deve tentar agradar a todos e nem a equipe de desenvolvimento do projeto;

- **Usar uma nova idéia:** o *Game Designer* deve tentar não seguir os padrões dos jogos no mercado, pois o público local prefere o que vem de fora, portanto terá que conquistá-los pela inovação. Acostumar-se a ter idéias próprias, mesmo que pareçam ruins. Grandes jogos partiram de idéias aparentemente estúpidas;

- **Ser Flexível:** O jogo não deve estar amarrado a um sistema específico ou a resoluções de telas muito grandes, a maioria dos possíveis usuários podem não ter uma máquina como as máquinas da equipe de desenvolvimento;

- **Projetar seu jogo para o futuro:** O jogo deve ser criado para executar nas placas padrão do mercado na época de seu lançamento e não na época de sua idealização. A informática avança muito rápido mas os rumos nem sempre estão claros. Cabe a equipe avaliar muito bem isso.

- **Pensar em séries, seqüências e expansões:** é importante guardar algumas boas idéias para a versão II do game. Grandes sucessos sempre pedem uma segunda versão para se manterem vivos, além disso, boa parte dos códigos criados para a primeira versão podem ser reaproveitados em uma versão seguinte, por isto a importância de se criar um game, orientado aos conceitos de re-usabilidade de código;

- **Conteúdo é tudo:** oferecer bons gráficos, sons e jogabilidade. Garantir que o jogo será divertido. Um bom exemplo disto é o game de tiro "*Duke Nukem*" que, pela primeira vez, coloca o jogador em situações divertidas, como matar um monstro que esteja urinado no banheiro ou gastar um pouco de dinheiro com dançarinas em um salão de *striper* que nada tinha haver com a trama principal do jogo;

- **Dar objetivos ao jogador:** jogos sem objetivos são odiáveis. É importante deixar o jogador sempre sentir que está indo para algum lugar que possa cumprir algo e não vagando por um mundo sem rumo. Premiar o jogador que cumprir objetivos intermediários e ter sempre em mente: ninguém gosta de não saber o que está fazendo ou para onde está indo;

Certamente essa lista está em constante mudança, pois novas práticas poderão ser descobertas a qualquer momento e novas tecnologias poderão causar ajustes nessa estrutura, mas cabe sempre ao Game Designer a palavra final e o bom senso na hora de desenvolver sua idéia.

Além dessas regras, conhecer o público-alvo do jogo é peça fundamental. Não é necessário dizer que cada tipo de jogo possui um público-alvo diferente, com anseios diferentes, mas de forma geral, alguns itens são comuns a quase todos os tipos de jogos. Para o levantamento desses itens duas perguntas devem ser feitas:

- O que os jogadores querem de seus jogos?
- O que os jogadores esperam de seus jogos?

A primeira pergunta fala dos anseios dos jogadores. O que eles realmente desejariam que tivesse em um jogo? Que elementos fariam esses usuários jogar repetidamente àquele jogo? A segunda pergunta tem um foco diferente, nela o *Game Designer* busca saber o que obrigatoriamente um jogo deve ter. São detalhes, muitas vezes imperceptíveis aos usuários quando existem, mas altamente notados quando não estão presentes no jogo.

As respostas a essas perguntas são as mais diferenciadas possíveis, porém, na compilação desses dados, chega-se a um consenso geral. E para a primeira pergunta, tem-se o seguinte resultado:

- **Desafio:** é a verdadeira motivação de um jogo. Os desafios sempre servem como experiência de aprendizado e geram emoção ao serem superados;

- **Socializar:** jogos provocam experiências sociais com amigos e familiares. Os usuários querem se reunir com amigos para jogar;

- **Experiência Solitária:** o oposto do item anterior também é verdade. Muitos jogadores gostam de jogar sozinhos, para passar o tempo e vencer desafios que necessitam de uma maior concentração;

- **Respeito:** os jogadores gostam de ganhar respeito ao vencerem desafios. Ver seu nome no topo do ranking causa orgulho e disputas acirradas na jogatina;

- **Experiência Emocional:** todos procuram algum tipo de emoção em um jogo, seja a adrenalina e a tenção de "*Quake*", seja o terror e o suspense de "*Phantasmagoria*", seja o espírito aventureiro de "*Sonic*" ou ainda o heroísmo do "*Homem-Aranha*";

- **Fantasia:** os jogadores também querem ser mais que simples mortais. Querem voar, nadar, matar alienígenas, combater o crime e até comandar a melhor seleção do mundo na Copa e os games podem proporcionar isto e muito mais, cabe ao *Game Designer* tornar isto possível;

Já para a segunda pergunta, os resultados são mostrados a seguir:

- **Mundo Consistente:** jogadores devem entender com clareza, o resultado de suas ações, como por exemplo: “Por que errei o tiro?”, “Por que meu soco não atingiu o adversário?”, etc.

- **Entender os Limites do Mundo:** o jogador também deve entender, com perfeição, o que pode e não pode fazer. Exemplo: o Mario nunca vai pedir a rendição do Koopa;

- **Direção:** deixar visível a direção onde o jogador deve seguir. Não de maneira óbvia, mas com dicas consistentes. Ninguém gosta de ter que vasculhar enormes labirintos e mundos virtuais para encontrar algo perdido em lugar completamente ignorado;

- **Cumprir tarefas progressivamente:** faz-se isto dando sub-objetivos ao jogador. Gratificar o jogador enquanto ele caminha para um objetivo maior;

- **Imersão:** fazer o jogador entrar em seu mundo, em todos os sentidos. Quem jogou “Super Mário” vai sempre lembrar o jogo apenas escutando sua música ou vendo algum cano verde por ai;

- **Falha:** deve-se fazer o jogador fracassar algumas vezes, jogos fáceis são desprezados rapidamente, mas o cuidado com os exageros, dificuldade extrema também afasta possíveis jogadores. A dificuldade deve ser aumentada gradativamente;

- **Não gostam de repetição:** não oferecer desafios iguais ao jogador. A repetição torna o jogo cansativo. Fazer o usuário iniciar sempre do início quando cometer uma falha é algo impensado nos jogos atuais;

- **Não deixar trancado:** cuidado com os labirintos. Deve-se sempre criar uma saída plausível. O jogador não deve ficar preso em uma fase, por que não ativou algum evento em fases anteriores;

- **Querem fazer e não ver:** não inserir *cutscenes* (vídeos não jogáveis) longos, tal ação tira a interatividade do jogo. Os jogadores querem interagir e não assistir. Se o *cutscene* for muito necessário, o jogador deverá ter disponível a opção de sair dele e ir diretamente ao início da jogatina;

- **Labirintos:** que não sejam óbvios demais, nem cansativos demais, sua presença é importante, mas deve ser bem balanceada. Ter cuidado com a coerência com o tipo de jogo que está sendo criado, o jogador, não deve, por exemplo, ser obrigado a resolver um problema de matemática, para abrir uma porta, em um game de Ação;

Depois de conhecer bem a sua equipe de desenvolvimento, as regras para um bom jogo e os desejos gerais de seu público-alvo, o *Game Designer* precisa iniciar sua criatividade, pensando qual o tipo de jogo ele quer apresentar ao público. Para isso, deve conhecer bem os tipos de jogos existentes no mercado. Essa classificação de tipos de jogos não é padronizada e cada autor lança uma lista própria, porém, de regra geral, temos os seguintes tipos:

- **Estratégia em Tempo Real:** É caracterizado por jogos de batalhas, onde o jogador administra exércitos, suprimentos, acampamentos e todo tipo de recurso para evolução e superação de um grupo adversário. Na figura 1.1, vemos o jogo *Age of Empires* que se encaixa perfeitamente nesse tipo. Outros exemplos: "*Age of Mitology*"; "*Pharaoh*"; "*Age of Wonders*" (este último feito em Delphi);

Figura 1.1. Age of Empires

- **Estratégia Simples:** São jogos onde não há batalha, o jogador precisa de muito raciocínio lógico avançar neste tipo de game, montando cidades ou componentes

destas cidades. Na figura 1.2, vemos o jogo *SimCity*., um clássico entre os jogos de estratégia simples. Outros exemplos: "*SimTower*"; "*Tycoon*"; "*The Sims*";

Figura 1.2. Sim City

- **RPG:** É a abreviação do termo *Role Playing Game*, que já foi traduzido de várias formas em nossa língua. É um estilo onde você desenvolve o personagem do jeito que preferir, interpretando um papel, em um mundo virtual. A presença de diálogos entre os personagens é uma marca importante desse tipo de jogo. O estilo RPG possui algumas variantes. Entre elas estão o Live Action, onde o jogador precisa literalmente vestir as roupas de seu personagem; o MMORPG (Multijogador Massivo Online), onde o jogo não possui um fim e permite a milhares de jogadores criarem personagens em um mundo virtual dinâmico ao mesmo tempo na Internet, uns contra os outros; e o CORPG (Cooperativo/Competitivo Online), semelhante ao MMORPG, porém os jogadores trabalham em conjunto para construir o mundo virtual, contra as adversidades apresentadas pelo computador. Na figura 1.3 vemos uma das versões do exemplo mais conhecido desse tipo de jogo: *Final Phantasy*. Outros exemplos: "*Phantasy Star*"; "*Dungeons & Dragons*"; "*Technoclash*."; "Os Federa 2"; "Os Federa 4 e a Máquina do Tempo". Esses dois últimos foram desenvolvidos na região do vale do São Francisco.

Figura 1.3. Final Phantasy

Tiro: São games, em quase sua totalidade, jogados na primeira pessoa, ou seja, o que o jogador vê na tela é a mesma cena que os olhos do personagem vêem no mundo virtual. Neste tipo de jogo, o jogador usa armamentos, com ou sem mira, e manda bala em tudo que vê pela frente. Ganhou fama com os games em 3D. Um grande sucesso de jogos de tiro foi o jogo DOOM, visto na figura 1.4. Outros exemplos: "*Wolf3D*"; "*Quake*"; "*The House of Dead*"; "*Duke Nuken 3D*"; "*Counter-Strike*";

Figura 1.4. Doom

Ação: São jogos de terceira pessoa, ou seja, o usuário vê todo seu personagem como se assistisse a um filme, onde o jogador precisa socar, atirar, lutar, conseguir suprimentos, armas, para cumprir missões com rapidez. Utiliza-se de elementos reais em um mundo virtual. Um dos jogos mais lembrados de todos os tempos é de ação, o clássico *Street of Rage*, visto na figura 1.5, com seus personagens mais conhecidos: Axel e Blaze. Outros exemplos: "*Ninja Gaiden*"; "*Shinobi*"; "*Final Fight*"; "*Alien* 3";

Figura 1.5.Street of Rage

Aventura: São similares aos jogos de Ação, porém com uma diferença básica: utiliza-se de personagens fantasiosos e elementos sem semelhança com o mundo real, para cumprir sua missão. Geralmente possuem personagens não humanos, ou de desenhos animados. Algumas vezes são jogos com uma carga forte de terror. Entre esses jogos está o inesquecível "Super Mario Bros", visto na figura 1.6. Outros exemplos: "*Sonic*"; "*Alex Kidd*"; "Mônica na Terra dos Monstros"; "*Castle of Ilusion*"; "*Monkey Island*"; "*Phantasmagoria*"; "*Night Trap*"; "Os Federa 5 e as Gangs de Rua", esse último produzido no vale do São Francisco.

Figura 1.6. Super Mário Bros

Puzzles: São jogos simples, rapidamente resolvidos com raciocínio lógico e/ou velocidade nos dedos. Geralmente são pequenos e distribuídos gratuitamente pela Internet. Um jogo bastante conhecido desse tipo é o *Tetris*, visto na figura 1.7. Outros exemplos: "*Columns*"; "*Pipe Dream*"; "*Frozen Bubble*"; "*Blockout*";

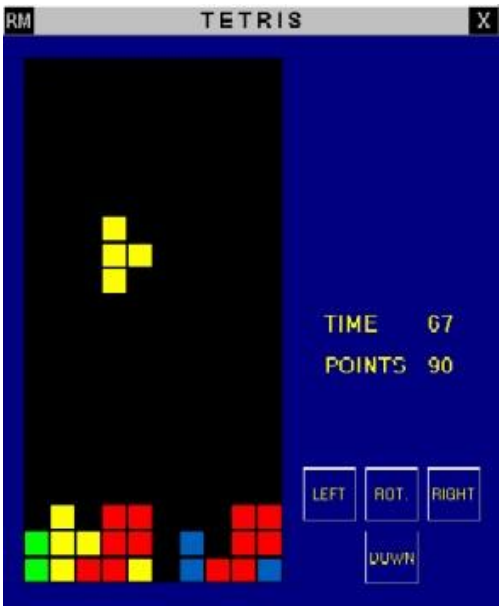


Figura 1.7. Tetris

Esporte: Como o próprio nome já diz, são jogos que simulam algum esporte, com exceção das modalidades de automobilísticos e de luta, pois estes possuem uma classificação própria. A figura 1.8 mostra o jogo *Fifa Soccer*, um simulador de jogos de futebol. Outros exemplos: "*Pro Evolution Soccer*"; "*Tennis*"; "*Califórnia Games*"; "*Olympic Gold*"; "*Golf*"; "*NBA Street*";

Figura 1.8. Fifa Soccer

Corrida: São jogos onde o jogador vai competir com outros concorrentes em uma situação onde serão dadas voltas em um circuito fechado e quem conseguir efetuar um número específico de voltas em um menor tempo, será o campeão. Em sua maioria, o jogador assume um carro, mas também pode ser um cavalo, um caminhão, uma nave, ou até mesmo a pé. Um grande exemplo desse tipo de jogo é o *Need for Speed*, o qual podemos ver na figura 1.9, neste jogo, o jogador assume o controle de um super carro e terá que vender outros super carros, em uma disputa acirrada e muito divertida. Outros exemplos: "Super Mônaco GP"; "*Nascar Race*"; "*Rock'N Roll Race*"; "*Daytona*"; "Super Mario *Kart*";

Figura 1.9. Need For Speed

Jogos de Mesa: São jogo que simulam jogos de tabuleiro ou de cartas, onde o jogador enfrenta o computador ou outro jogador em partidas de muito raciocínio lógico, utilizando-se das mesmas regras do jogo de tabuleiro real. É neste estilo que se encontra um dos jogos eletrônicos mais jogados da história, o Paciência, mostrado na figura 1.10. Outros exemplos: "*Chessmaster*"; "Damas"; "Gamão"; "*FreeCell*";

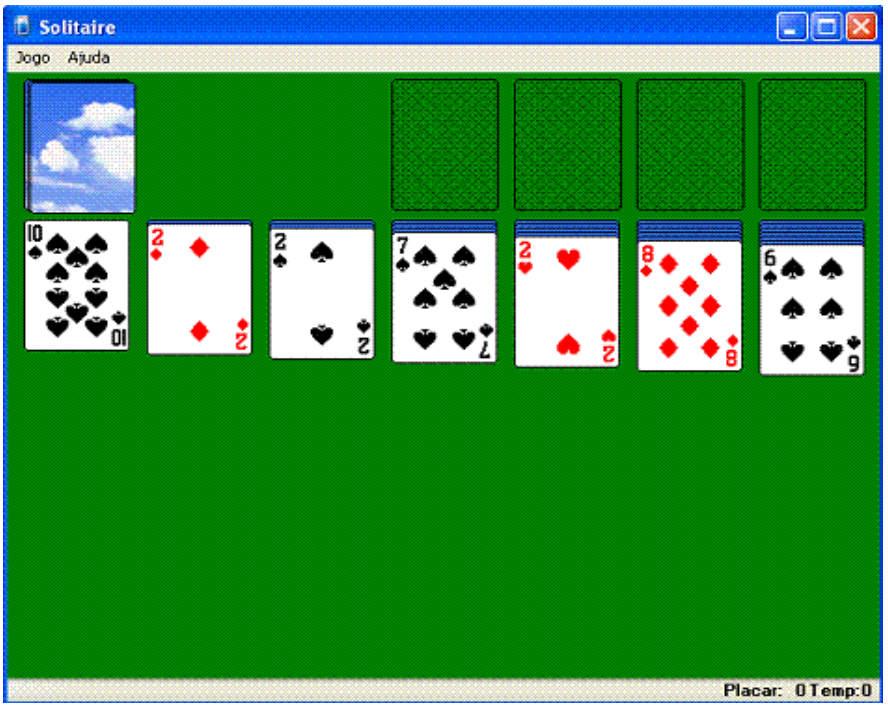


Figura 1.10. Paciência

Luta: São jogos onde o jogador escolhe um personagem e joga de cenário em cenário, (onde cada cenário é como um *ring* de luta, mesmo que não pareça um *ring*) enfrentando um a um, oponentes com habilidades específicas, mas similares. O maior exemplo desta categoria é o Street Fighter 2, onde o jogador pode escolher um entre 8 opções de lutadores, e depois enfrentará os demais e mais quatro personagens finais, considerados personagens do mal. Outros exemplos: "*Mortal Kombat*"; "*Eternal Champions*"; "*Soul Calibur*"; "*Fight Masters*"; "*Saint Seiya*: *Chapter Sanctuary*"; "X-*Men vs*. *Street Fight*"; "Os Federa vs. X-men". Esse último produzido no vale do São Francisco.

Figura 1.11. Street Fighter II

Nave: São jogos onde o jogador assume o comando de uma determinada nave, aérea ou espacial e segue destruindo inimigos em missões específicas, sem nunca abandonar essa nave, ou seja, o jogador é a própria nave. Muitas vezes essas naves têm designs improváveis e/ou vêm equipadas de utensílios especiais para ajudar nas missões, ou simplesmente são cópias fiéisdas aeronaves reais, como é o caso do jogo *Flight Simulator*, visto na figura 1.12, onde o jogador terá que executar todas as tarefas de um piloto real, sendo este jogo considerado um perfeito simulador de vôo. Outros exemplos: "*Space Tripper*"; "*Truxton*"; "*Desert Strike*"; "*River Raid*"; "*After Burner 3D*";

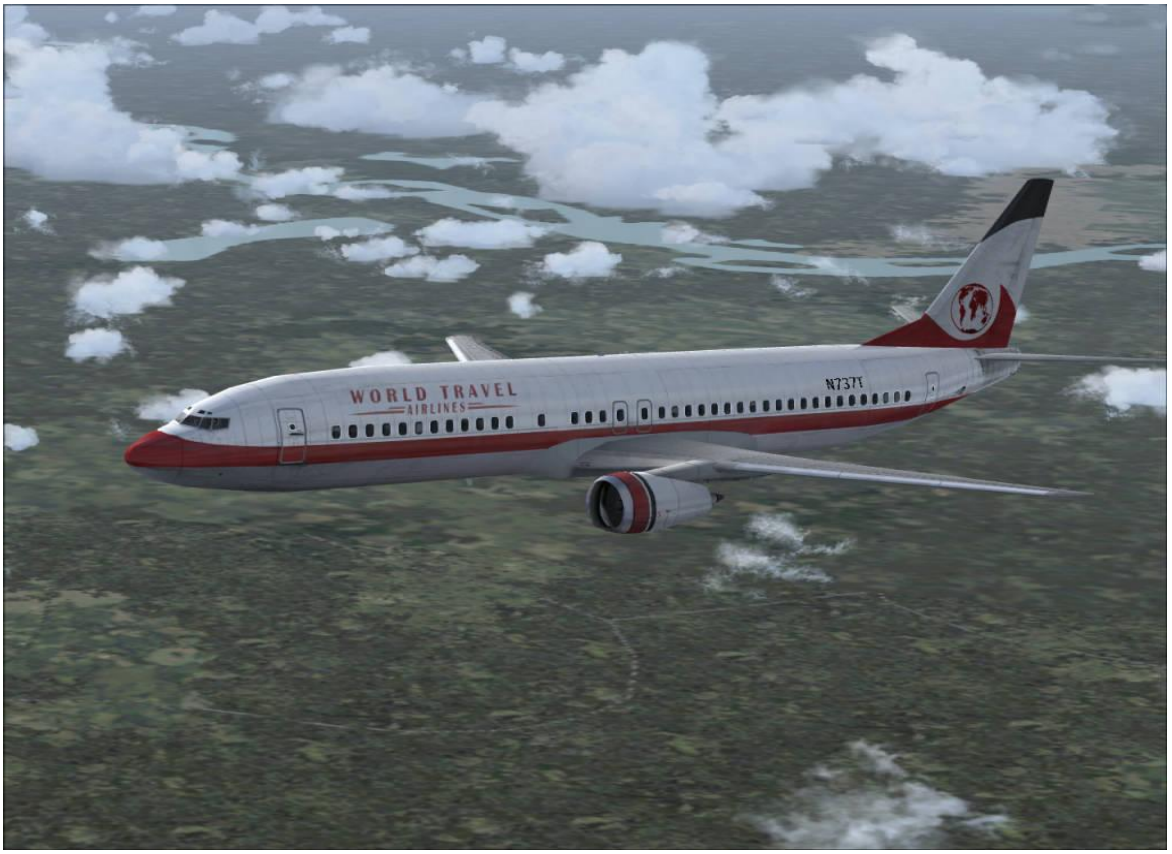


Figura 1.12. Flight Simulator

Cada um desses tipos tem suas especificações que devem ser tratadas bem a fundo pelo *Game Designer*. Além disso, o desenvolvimento de jogos eletrônicos também depende da plataforma a qual o jogo irá executar. O *Game Designer* poderá optar por máquinas de *arcade*, consoles portáteis, consoles de mesa, computador ou dispositivos móveis.

- **Arcade** – Na sua grande maioria, são máquinas dedicadas a execução de um único jogo, portanto, podem ser moldadas e plenamente adaptadas a esse jogo;

- **Consoles portáteis** – Apesar de serem máquinas criadas com objetivo exclusivo para jogos, elas possuem um hardware muito limitado e os recursos a serem utilizados nos jogos devem ser bastante simples, adaptados a esses sistemas;

- **Consoles de Mesa** – Assim como os consoles portáteis, são máquinas criadas com objetivo exclusivo para jogos, porém, com capacidade bem superior aos portáteis. Elas não podem ser adaptadas aos jogos, como os arcades, o que também é um fato limitador de recursos, mas possuem sistemas que valorizam a padronização e a portabilidade de seus jogos, além de terem sua arquitetura conhecida, onde o *Game Designer* poderá utilizar o máximo dos recursos disponíveis.

- **Computadores** – São máquinas "genéricas" onde o *Game Designer* não conhece plenamente os recursos das máquinas de seus usuários, o que torna o desenvolvimento um desafio. O desenvolvedor deve tentar buscar utilizar recursos das máquinas mais vendidas na época do lançamento e não na idealização do jogo. Como a evolução das máquinas é muito rápida, o desafio da equipe é muito maior em projetos de longa duração;

- **Dispositivos móveis** – Assim como os computadores, são máquinas "genéricas", mas com recursos muito limitados, por vezes, mais limitados do que os consoles portáteis. São máquinas como *tablets* e celulares, com recursos de hardware muito pequenos e onde o principal desafio é tentar imitar os jogos dos computadores, sem seus recursos disponíveis, além disso, a grande diversidade de modelos desses dispositivos obriga ao *Game Designer* a desenvolver seus jogos em máquinas virtuais que testam seu jogo em diversos tipos de modelo e fabricantes diferentes.

1.6. Introdução ao HTML5

O HTML, sigla em inglês que significa *Hypertext Markup Language* é uma linguagem de marcação, direcionada para a estruturação de documentos e apresentação visual desses documentos em um aplicativo conhecido como *browser* (navegador).

O HTML utiliza uma forma de "apontar e clicar" para seguir no conteúdo desejado, conhecido nos dias de hoje como link ou ancora. Surgiu da fusão dos padrões SGML e *HyTime* e foi desenvolvida por Tim Berners Lee (o mesmo que criou o WWW) para a composição e apresentação de documentos na web. A evolução cronológica das tecnologias pode ser organizada da seguinte forma: SGML, HTML 1.0, HTML x.x, XML, HTML 4.01, XHTML, HTML5.

Um documento estruturado é composto por conteúdo, que pode ser em formato de texto, imagens, vídeos, entre outros, e informação sobre como o conteúdo será apresentado. Essa informação é conhecida como *layout* do documento.

A apresentação do conteúdo ao usuário final pode ser feita de diferentes formas. Cada uma dessas formas existe com o intuído de se adaptar melhor a cada tipo de usuário e de equipamento que possa ser utilizado para se ter acesso a esse conteúdo, como por exemplo, computadores pessoais, *notebooks*, *netbooks*, celulares, *tablets*, etc.

A versão anterior ao HTML5, o HTML 4.0.1 foi aprovada em 1999 e desde então nada foi alterado. Essa realidade só veio a mudar em 2010, quando o HTML5 foi apresentado ao mundo, mesmo ainda estando em processo de aperfeiçoamento e ajustes. A necessidade de evolução surgiu com o aparecimento de novas tecnologias e dispositivos como os *smartphones* e os *tablets*, que até então estavam sendo supridos

por outras tecnologias ainda não tão difundidas e muito mais restritas a estes dispositivos.

O HTML5, cuja logomarca é mostrada na figura 1.13, traz importantes mudanças na estruturação básica de seus documentos e novas funcionalidades como semântica e acessibilidade. Fundamentalmente essas mudanças visam oferecer suporte para as mais recentes mídias, enquanto que por outro lado, mantém sua legibilidade para os desenvolvedores e consistência para os mais diferentes tipos de terminais.

Figura 1.13. Logomarca do HTML5

1.6.1. Novidades do HTML5

Várias novidades foram embutidas ao HTML5, todos com a finalidade de facilitar a compreensão e a manutenção do código. Algumas dessas novidades são simplesmente a evolução natural de instruções, chamadas de *tags*, já existentes em versões anteriores, como por exemplo, o comando , que não influencia diretamente no visual da página, mas é importante para a semântica do código, porém, entre as novidades mais importantes estão à criação de novas API's, como por exemplo, a API para criação de gráficos bidimensionais, o controle embutido de conteúdo multimídia, o aprimoramento do uso off-line, melhoria na depuração de erros e o surgimento de novas *tags*, que padronizam a maneira de se publicar conteúdo. Entre essas *tags*, tem-se:

Estruturais (sintetizados na figura 1.14)**:**

- <header> – cria o cabeçalho da página ou de uma seção (diferente da *tag* <head>);
- <nav> – o grupo de links que formam a navegação, seja os links do menu principal do site ou relacionados ao conteúdo da página;
- <section> – separa cada seção do conteúdo;
- <article> – coloca um item do conteúdo dentro da página ou da seção;
- <footer> – cria o rodapé da página ou de uma seção;
- <aside> – conteúdo relacionado ao artigo (muito útil em blogs por exemplo);
- <!DOCTYPE html> – define o *doctype* de maneira curta e simples;

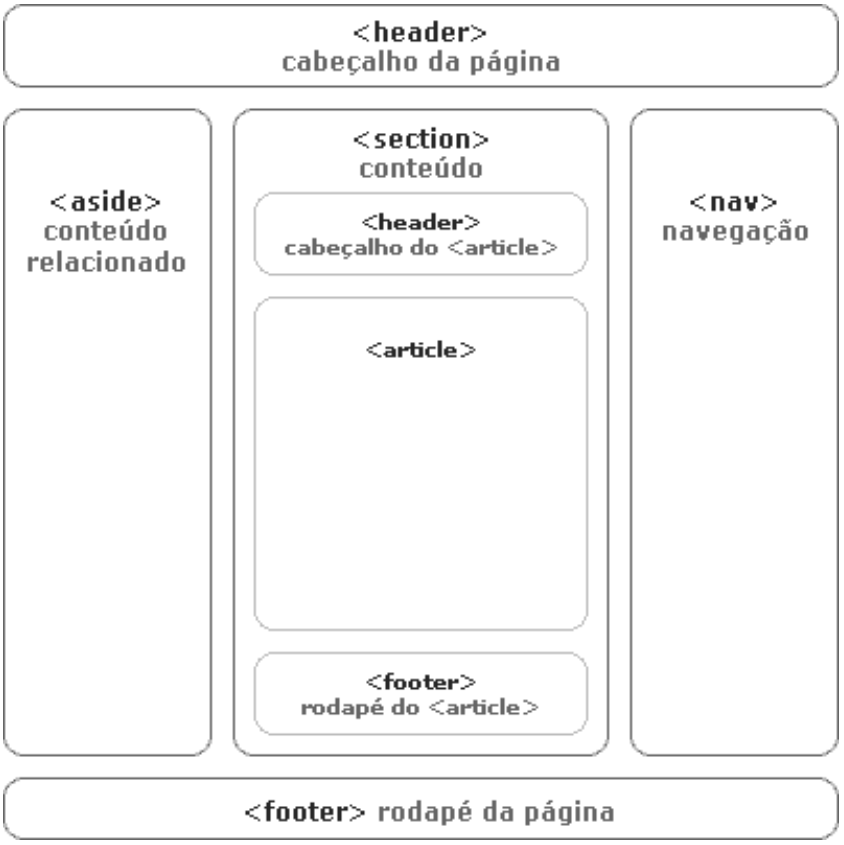


Figura 1.14. Estrutura do site em HTML5

Conteúdo:

- <figure> – utilizado para associar uma legenda a uma imagem, vídeo, arquivo de áudio, objeto ou iframe;
- <legend> – define o conteúdo da legenda que será associada pela *tag* <figure>;
- <canvas> – por meio da API gráfica do HTML5, renderiza imagens 2D dinâmicas que poderão ser usadas em jogos, gráficos, simulações, etc;
- <audio> e <video> – utilizados para a tecnologia streaming (transmissão on-line de áudio e vídeo);
- <dialog> – junto com as *tags* <dt> e <dd> é usada na formatação de diálogos;
- <time> – representa hora e/ou data;
- <meter> – usado para representar medidas, como por exemplo, distâncias, armazenagem em disco, entre outras;

Entre as melhorias do HTML5, também ocorreu à substituição ou eliminação de algumas *tags* presentes em versões anteriores. Algumas delas foram retiradas porque sua função era apenas visual e foram substituídas por uma declaração no CSS (*Cascading Style Sheets*), como por exemplo: <basefont>, <big>, , <font>,<s>, <strike>, <tt> e <u>. Outras foram completamente excluídas por terem um efeito negativo na acessibilidade do site, como por exemplo: <frame>, <noframes> e <frameset>.

Embora consideradas antigas, as *tags* <b> (utilizada para negrito em palavras) e <i> (utilizada para itálico) ainda são reconhecidas e renderizadas para fins de formatação, mas devem ser substituídas, sempre que possível, pelas *tags* e , respectivamente.

Também foram excluídos alguns atributos de algumas *tags*, seja porque caíram em desuso pelos desenvolvedores web ou porque podem ser substituídos semanticamente por declarações no CSS para definir o visual dos elementos na página. Os principais atributos retirados são:

- background em <body>;
- target no elemento <a>;
- align nos elementos <table> e demais tags de tabelas, <iframe>, <img>, <input>, <hr>, , <p>, entre outros;
- noshade e size em <hr>;
- bgcolor nos elementos de tabela e no <body>;
- height em <td> e <th>;
- border em <table> e <object>;
- cellpadding e cellspacing em <table>;
- width nos elementos <hr>, <table>, <td>, <th> e <pre>;
- hspace e vspace em <img> e <object>;

O HTML5 deverá eliminar a necessidade de *plug-ins* para aplicações multimídia em navegadores web. Muitos cientistas da computação consideram a tecnologia como um fortíssimo concorrente ao Microsoft *Silverlight*, ao Adobe *Flash* e ao Sun *JavaFX*. É provável que esses, e outros concorrentes do HTML5, precisarão evoluir rapidamente para conseguir manter-se no mercado ao mesmo nível em que se encontram hoje.

Em sua avaliação do co-diretor de ferramentas da Mozilla, Ben Galbraith, comenta que as tecnologias viabilizadas pelo HTML5 como o Canvas para desenhos 2D e o armazenamento de conteúdos no desktop, permitirão que "usemos mais o browser do que nunca".

Após uma década sem modificações, o modelo de escrita de páginas na web passa por uma forte transformação. O HTML5 oferece aos desenvolvedores uma experiência de criação totalmente diferente e embora ainda exista um longo caminho para ser finalizado, muitos *browsers* importantes, como Internet Explorer 9, Opera, Safari 4 e Chrome já implementaram uma grande parte da linguagem, incluindo as *tags* de vídeo e suporte à tecnologia Canvas. Com a evolução da linguagem, os navegadores passam de simples exibidores de conteúdo para renderizadores de páginas web.

1.7. Eestudo de Caso: Um Jogo de Nave

Para o estudo de caso do laboratório, será desenvolvido um jogo de nave, utilizando a IDE (*Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento) de desenvolvimento de jogos e *engine* GameMakerHTML5, visto em sua tela inicial na figura 1.15.

O Game Maker é um IDE (ou motor) de programação de jogos, também chamado de GM. Foi desenvolvido pela Yoyo Games e possui suporte a uma linguagem de script, chamada GML, muito familiar para desenvolvedores de C e principalmente de Delphi, linguagem onde o mesmo foi desenvolvido. O fato de ter sido desenvolvido em Delphi faz com que o mesmo tenha apenas versões para o sistema operacional Microsoft Windows.

O Game Maker foi desenvolvido pelo Prof. Dr. Mark Overmars, que lançou a primeira versão do GM em 15 de novembro de 1999. Nas primeiras versões, o motor era chamado de Animo e seu principal foco era a criação de animações em 2D, mas como alguns de seus utilizadores usavam o programa com o propósito de criar jogos, e não animações, Overmars decidiu mudar seu nome para Game Maker.

Mark Overmars também é o fundador da *Game Maker Community* ou Comunidade do Game Maker, onde administra um fórum que usa a tecnologia da *Invision Board*. Nesta comunidade são exibidos e discutidos elementos a serem utilizados no desenvolvimento de jogos, como por exemplo, gráficos, sons, vídeos, músicas, além de jogos em progresso, questões sobre o Game Maker e até mesmo sobre a comunidade em si. Além da comunidade, Overmars também fundou a Yoyogames. Site direcionado apenas para jogos desenvolvidos no Game Maker. No site os usuários podem mostrar suas criações para jogadores e outros membros que por sua vez, podem comentar um jogo, classificá-lo em uma escala de pontuação que varia de uma até seis estrelas, e enviar um *feedback* para o criador sobre o jogo.

1.7.1 Característica do GM

O Game Maker pode criar quaisquer tipos de jogos, sejam em 2D ou 3D, como também jogos *multiplayers* on-line e softwares em geral, principalmente animações e simulações.

Todos os recursos a serem utilizados em um projeto são organizados em pastas geradas dentro do programa, além disso, o GM inclui pequenos programas para auxiliar o desenvolvedor a criar seus próprios recursos (*resources*), como editores de imagens, sons, animações, scripts e fases. A ferramenta também permite salvar os recursos

criados para que possam ser utilizados em outros jogos (re-usabilidade) ou fora do programa, como também importar ações adicionais para estender as funções do programa, aumentando assim o poder da IDE.

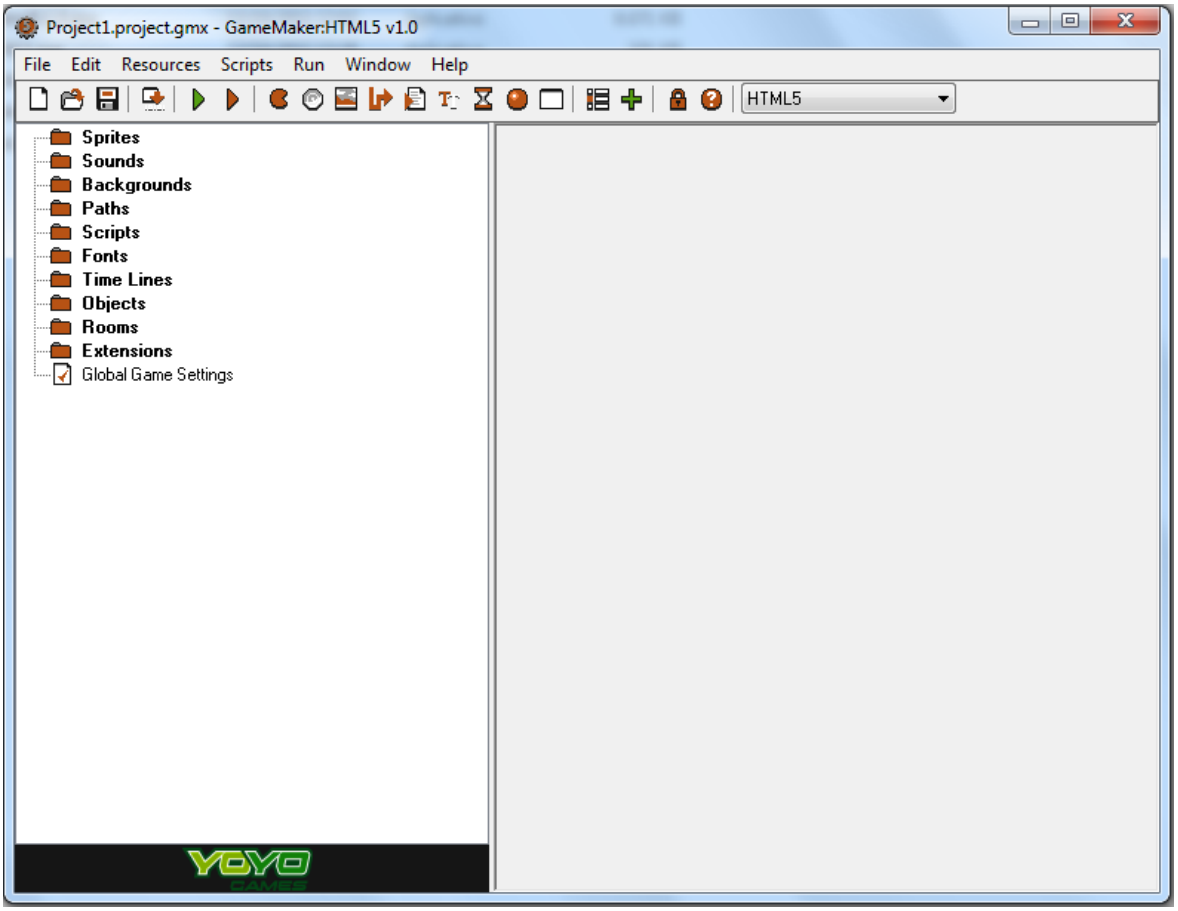


Figura 1.15. Game Maker HTML5

1.7.2. Passos para a Criação do Jogo

Como já foi dito, o primeiro passo para a criação de um jogo é o seu planejamento. Como o estudo de caso trata de um jogo muito simples pequeno, seu planejamento também será bem sucinto.

O jogo será chamado de Space Attack e só poderá ser jogado por um único usuário. Trata-se de um jogo de nave em 2D, formado por quatro telas. A tela inicial, onde o jogador verá a logomarca do jogo e irá pressionar *enter* para seguir à segunda tela. A segunda tela trata da única fase jogável do *game*, que terá um *scroll* de tela da direita para a esquerda. Nela, o usuário poderá controlar uma nave em quatro direções (cima, baixo, esquerda e direita) utilizando-se das teclas de direcionamento do teclado e poderá disparar munição, utilizando a barra de espaço. O jogador terá que destruir o máximo de inimigos colidindo a munição disparada com esses inimigos. Os inimigos surgirão à direita da tela e seguirão para a esquerda, tentando colidir com a nave do usuário. Caso o jogador consiga destruir uma quantidade de 20 inimigos, sem esbarrar nos mesmos pelo menos três vezes, seguirá para a terceira tela, onde o jogador será

declarado campeão. No caso da nave esbarrar três vezes em inimigos, será encaminhado para a quarta e última tela do projeto, onde será declarado perdedor do jogo. A primeira, terceira e quarta telas do jogo são meramente estáticas.

O prazo pra conclusão de jogo é estimado em 180 minutos, sendo 30 minutos para análise dos recursos, 30 minutos para a geração das telas, 60 minutos para programação dos eventos e 60 minutos para testes e correção de erros.

O custo do projeto é um valor simbólico, por si tratar apenas de uma demonstração acadêmica.

1.7.2.1. Recursos Utilizados

Todos os recursos utilizados em um jogo são chamados de *resources*. No Space Attack serão utilizados:

- Imagens (*Sprites*):

Figura 1.16.Nave controlada pelo usuário na posição normal

Figura 1.17. Nave controlada pelo usuário na posição de descida

Figura 1.18. Nave controlada pelo usuário na posição de subida.

A nave do usuário, mostrada nas figuras 1.16, 1.17 e 1.18 em todas as possíveis posições, não poderá sair do campo visível da tela e terá os movimentos norte, sul, leste, oeste.

Figura 1.19. Nave inimiga número 1

Figura 1.20. Nave inimiga número 2

Figura 1.21. Asteróide inimigo

Cada um dos inimigos, mostrados nas figuras 1.19, 1.20 e 1.21, só terão movimentos da direita para a esquerda e deverão ser apagados da memória, quando saírem da área visível da tela.

Figura 1.22. Munição da nave amiga.

Quando a munição da nave amiga, mostrada na figura 1.22 colidir com um dos inimigos, este explodirá. A imagem da explosão, pode ser vista na figura 1.23.

Figura 1.23. Explosão.

- Imagens (*Backgrounds*):

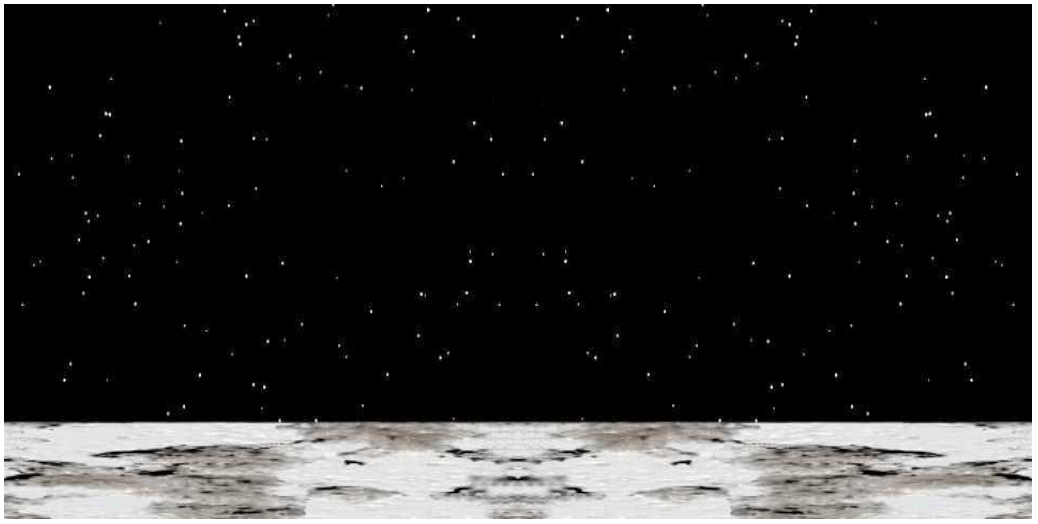


Figura 1.24. Tela da fase jogável.

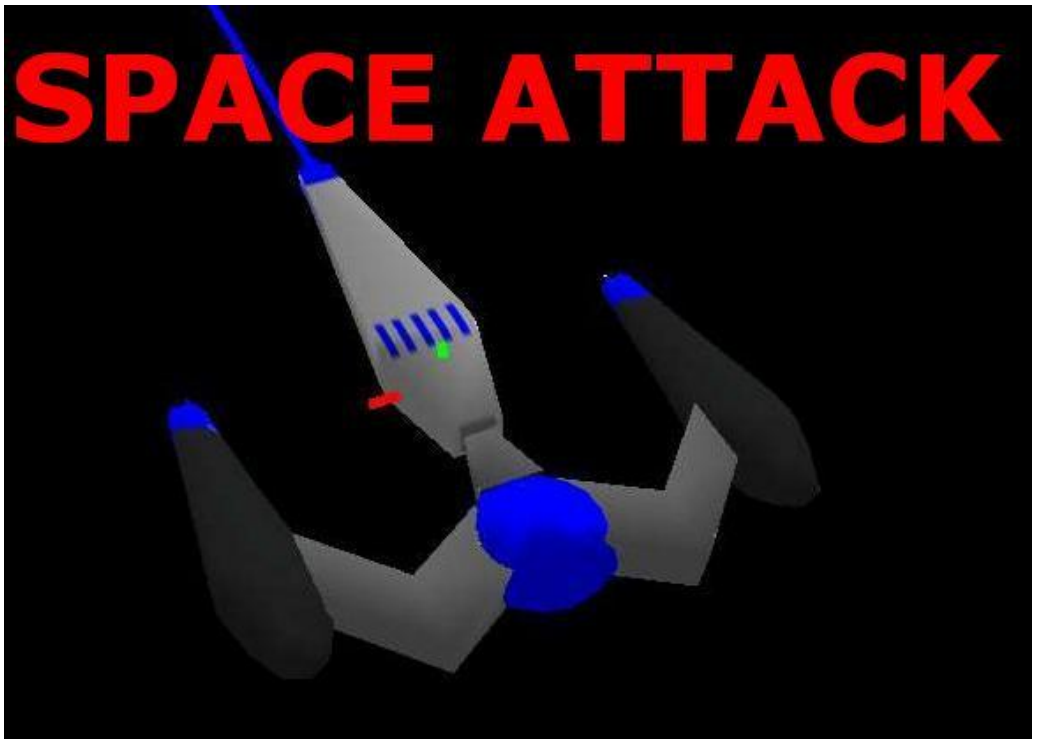


Figura 1.25. Tela inicial de abertura.

Figura 1.26. Tela da vitória.

Figura 1.27. Tela da derrota.

Com foi mencionado no planejamento, o jogo terá quatro telas. O visual de cada tela é mostrado nas figuras 1.24, 1.25, 1.26 e 1.27.

- Sons

  Para o jogo, foram selecionados sons da base de dados de sons do Maker. Os escolhidos foram:

  - Desert → como música de fundo para a tela de abertura;
  - Hurry Up → como música de fundo para a tela jogável do jogo;
  - Item01 → como música de fundo da tela de derrota;
  - Fanfare03 → como música de fundo da tela de vitória;
  - Explosion1 → como som emitido pela colisão da munição ou da nave do usuário, contra uma nave inimiga;
  - Thunder3 → como som emitido pelo disparo de uma munição pelo usuário;

Todos esses recursos devem ser importados antes do início da programação dos códigos. Os códigos serão feitos com base na orientação a objetos, onde serão criadas classes, conforme mostrado no diagrama de classes, na figura 1.28.

- Nave_amiga, que será controlada pelo usuário;
- Nave_inimiga, que será superclasse de cada uma das classes das naves inimigas;
- Nave-inimiga01, Nave-inimiga02 e Asteroide, são as classes que irão gerar os diferentes inimigos do usuário. Essas classes herdarão atributos da classe Nave_inimiga;
- Fabrica_Inimigos, que irá produzir incontáveis inimigos no jogo;
- Controle_pontos, que verificará se o jogador já atingiu os pontos suficientes para ser campeão;
- Construtor_jogo, onde serão declaradas todas as variáveis globais do jogo;
- Transicao_tela, que executará as transições de tela entre as telas não jogáveis do jogo;

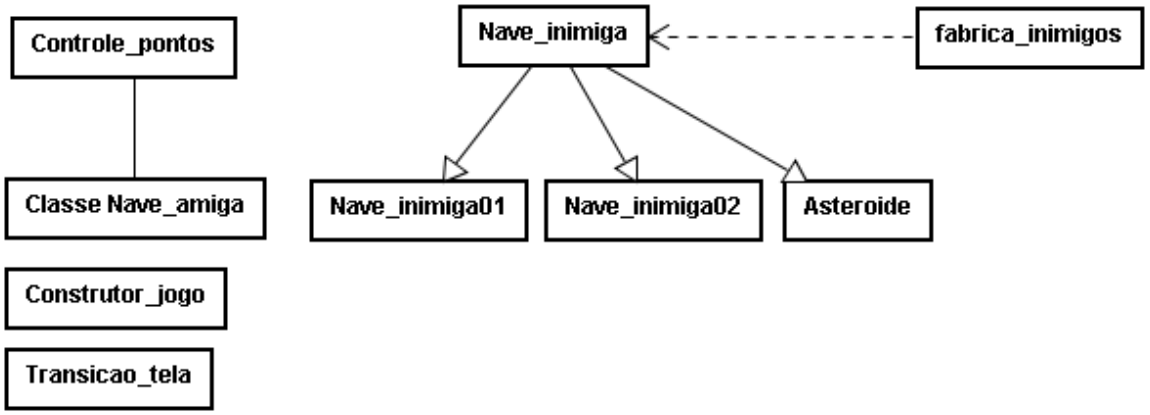


Figura 1.28. Diagrama de classes do jogo

1.8. Considerações Finais

Foi apresentada neste trabalho a nova ferramenta de desenvolvimento Web, intitulada HTML5, onde diversos renomados cientistas da área, afirmam ser o padrão do futuro. Também foi apresentado todos os passos para um desenvolvimento de jogos eletrônicos utilizados por empresas desenvolvedoras, culminando no desenvolvimento de um jogo completo para a nova tecnologia de programação HTML5.

Com este trabalho também conhecemos a IDE de programação GamemakerHTML5, que facilita o desenvolvimento de softwares, principalmente jogos, para o padrão Web HTML5.

Com a escassez de bons profissionais qualificados para o desenvolvimento de jogos eletrônicos no mercado, a iniciativa deste trabalho é de grande importância, pois é uma oportunidade para pessoas, que querem ingressar nesse meio, a dar o primeiro passo no difícil e fascinante universo de desenvolvimento de *games*.

1.9. Referências

ABRAGAMES (2012) – Associação Brasileira de Desenvolvedores de Jogos eletrônicos. Site organizacional. Disponível em http://www.abragames.org. Acessado em 08 de março de 2012.

BRAUN, Daniela. (2012) **HTML 5: conheça a linguagem que vai revolucionar sua navegação na web.** Artigo eletrônico, 2010. Disponível em http://idgnow.uol.com.br/internet/. Acessado em 18 de fevereiro de 2012.

FERNANDES, Maicris, (2002) **Programação de Jogos com Delphi usando Direct X**. Editora Relativa, Florianópolis, 2002.

FIGUEIREDO, Roberto T., FIGUEIREDO, Carla B. C**.** (2011) **WarGrafos – Jogo para auxílio na aprendizagem da disciplina de Teoria dos Grafos**. Artigo publicado no Simpósio Brasileiro de Games e Entretenimento Digital. Anais do Evento. Salvador, 2011.

PERUCIA, Alexandre S., BERTHÊM, Antônio C., BERTSCHINGER, Guilherme L., MENEZES, Roberto R. C. (2005) **Desenvolvimento de Jogos Eletrônicos – Teoria e Prática.** Editora Novatec, São Paulo, 2005.

SARTI, Erika. (2009) **Introdução ao HTML5**. Artigo eletrônico, 2009. Disponível em http://www.infowester.com/introhtml5.php. Acessado em 20 de fevereiro de 2012.

YOYO GAMES. (2012) Site oficial da empresa de desenvolvimento. Disponível em http://www.yoyogames.com/. Acessado em 22 de fevereiro de 2012.

Capítulo 2

Desenvolvimento de Hardware utilizando SystemVerilog

Rômulo Calado Pantaleão Camara

Abstract

This paper presents the main steps for anyone who aspires to start developing hardware using the hardware description languages SystemVerilog. The process will be detailed since the tools installations, through the details of language, until the development of examples with different levels of complexity. It will be present direct verification forms to assist the designer in the correct description of the hardware.

Resumo

Este trabalho traz os principais passos para quem almeja iniciar o desenvolvimento de hardware utilizando a linguagem de descrição de hardware SystemVerilog. O processo será detalhado desde a instalação da ferramenta necessária, passando pelo detalhamento da linguagem, até o desenvolvimento de exemplos com diferentes níveis de complexidade. Serão apresentadas, também, formas de verificação direta para auxiliar o desenvolvedor na descrição correta do hardware.

2.1. Considerações iniciais

A abstração é uma ferramenta do mundo real que permite utilizar máquinas e outros objetos complexos, como automóveis, computadores, braços robóticos, com apenas algumas horas de treinamentos. De acordo com [1], abstração é o processo ou resultado de uma generalização por redução do conteúdo de uma informação, conceito ou fenômeno observável. Pode-se observar que esse conceito é bastante utilizado pelos cientistas da computação para facilitar o uso de dispositivos eletrônicos por qualquer pessoa. A maioria das pessoas, quando utilizam um computador para jogar, assistir um filme, acessar redes sociais e trabalhar, não sabe exatamente o que acontece dentro da máquina. Essa ferramenta também é utilizada pelos engenheiros especializados na descrição de hardware.

Em 1958, Jack Kilby construiu o primeiro circuito integrado flip-flop do mundo, utilizando dois transistores na indústria Texas Instruments. Até o final da década de 70 esses circuitos eram desenvolvidos através de desenhos a mão, transistor por transistor, fio por fio. Todavia, isso limitava o tamanho dos circuitos desenvolvidos, além de gerar

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

uma imensa quantidade de erros. Um desenho tem menor portabilidade, é mais complexo para compreensão e extremamente dependente da ferramenta utilizada para produzi-lo. Após o desenho dos pequenos circuitos, era necessário desenvolver simulações SPICE(*Simulated Program with Integrated Circuits Emphasis*) para testar se os mesmos continham algum erro. Simulações SPICE atualmente são utilizadas no desenvolvimento de bibliotecas de portas lógicas, ou seja, circuitos integrados muito pequenos. Essa diminuição de uso do SPICE se deve à abstração criada por pesquisadores para o desenvolvimento de circuito integrado.

No final da década de 70, surgiram as primeiras Linguagens de Descrição de Hardware: ISP (Processador de Conjunto de Instruções), desenvolvido na Carnegie Mellon University, e KARL, desenvolvido na Universidade de Kaiserslautern, ambos por volta de 1977. Porém, ISP era utilizada como uma linguagem para desenvolver simulações e não poderia ser utilizada para desenvolver códigos sintetizáveis. KARL era utilizada diretamente em nível de hardware como apoio para etapa de floorplaning, onde as portas lógicas são organizadas dentro de um espaço que será o chip.

A primeira Linguagem de Descrição de Hardware moderna surgiu em 1980 e foi desenvolvida pelo Departamento de Defesa dos Estados Unidos da América (DARPA) para documentar o comportamento de *Application Specific Integrated Circuits* (ASIC), que compunham os equipamentos vendidos às Forças Armadas americana, e denominava-se VHDL (*VHSIC Hardware Desciption Language*), sendo baseada na linguagem de programação Ada. Em 1987, após o sucesso inicial do uso da VHDL, a sua definição foi posta em domínio público, o que levou a sua padronização pelo IEEE.

Em 1985 foi apresentada Verilog, a primeira linguagem de descrição de hardware de domínio público pela empresa Gateway Design Automation. Alguns anos mais tarde, a empresa Cadence Design Systems adquiriu os direitos do simulador Verilog-XL, que se tornaria o padrão dos simuladores de Verilog e investiu para que essa linguagem se tornasse uma das mais utilizadas mundialmente. Foi padronizada pelo IEEE e sofreu algumas modificações para deixá-la mais usual e simples de ser utilizada.

Quando Verilog foi criado, o tamanho de um típico projeto de circuito integrado era na ordem de 10.000 portas lógicas. Contudo, após as Linguagens de Descrição de Hardware esse número saltou exponencialmente: podemos observar esses números a partir de um estudo feito pela Intel, que demostra que em 1958 existia o primeiro circuito com dois transistores e em 2003 lançou um microprocessador contendo 55 milhões de transistores. Em 2012 a mesma empresa desenvolveu um microprocessador com 2,6 bilhões de transistores. O gráfico 2.1 mostra a evolução da quantidade de transistores em um mesmo chip.

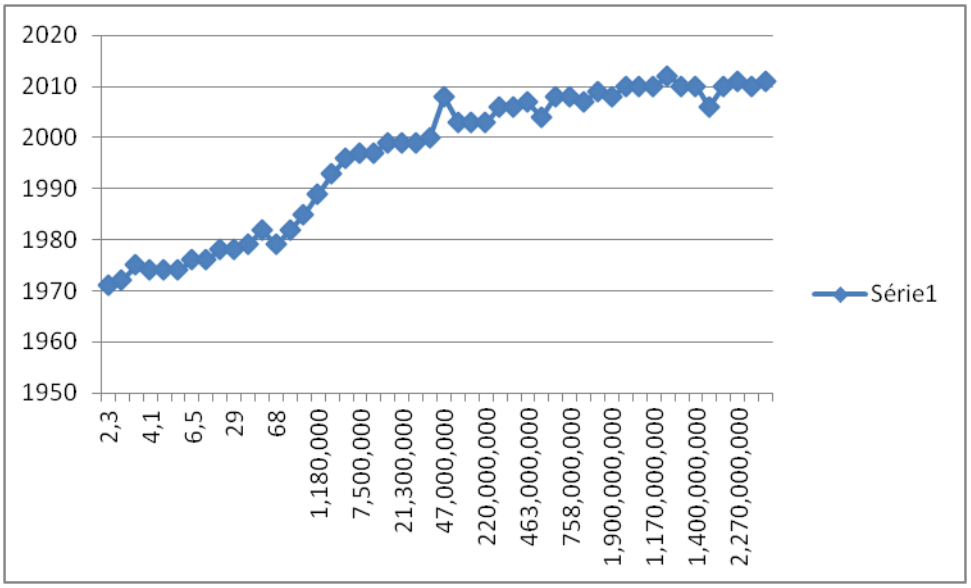


Figura 2.15. Evolução da quantidade de transistores num mesmo chip

Assim, podemos observar que devido à abstração alcançada pelas Linguagens de Descrição de Hardware, os circuitos integrados obtiveram um grande avanço na complexidade e velocidade de processamento.

Todavia, as linguagens também foram sendo modificadas para inserções de novos padrões de descrição, tornando a abstração das portas lógicas ainda mais forte. Em 2001 a IEEE retificou uma nova extensão para a linguagem Verilog com novos recursos, esperados ansiosamente pelos usuários. Foi desenvolvido um estudo minucioso e inseridas novas funcionalidades, como matrizes multidimensionais, variáveis automáticas entre outras. Atualmente, todas as Empresas *Electronic design automation* (EDA) suportam esse novo padrão de Verilog.

O avanço das HDLs teve principal suporte no avanço das linguagens de programação; novos paradigmas foram desenvolvidos e o avanço em Verilog deveria continuar. Nesse contexto, algumas empresas se uniram para desenvolver uma outra Linguagem de Descrição de Hardware que englobasse tudo o que Verilog já supria. Esse desenvolvimento tinha o objetivo de criar uma linguagens que pudesse ser utilizada não só no processo de *design*, mas também na Verificação Funcional, processo onde se verificam todas as funcionalidades de um chip que está sendo desenvolvido. Tal linguagem foi denominada de SystemVerilog.

A Introdução de *SystemVerilog* como uma nova linguagem de descrição de hardware e verificação funcional é motivada pela necessidade de ferramentas poderosas que facilitem a implementação de metodologias mais recentes. O objetivo principal da criação é unir todo o fluxo *front-end* de um projeto de hardware em uma única linguagem. O padrão IEEE 1800 ™*SystemVerilog* é a primeira linguagem de descrição de hardware e de verificação (HDVL) unificada da indústria. *SystemVerilog* é uma extensão importante do padrão IEEE 1364 ™linguagem Verilog, desenvolvido originalmente pela empresa Accellera para melhorar drasticamente a produtividade no projeto, com um número elevado de portas lógicas. Orienta-se principalmente para a implementação de chips e fluxo de verificação. Além disso, possui alguns recursos de modelagem a nível de sistema. As principais extensões criadas foram:

- Suporte a modelagem e verificação em nível de transação. *SystemVerilog* Direct Programming Interface (DPI) habilita a linguagem a fazer ligações a funções

C/C++/*SystemC*. Desse modo, torna-se a primeira linguagem baseada em Verilog a oferecer cossimulação eficiente com blocos *SystemC*, muito importante para fazer ligação da modelagem de sistema com o ambiente de verificação e a implementação RTL do chip;

- Um conjunto de extensões para atender as exigências de design avançado que utilizam o modelo de programação orientado a objetos. Interfaces de modelagem que aceleram consideravelmente o desenvolvimento de projetos, eliminando as restrições sobre as conexões de porta do módulo, dentre outros;
- Um novo mecanismo de apoio à verificação, chamado de *assertions*, permitindo uma metodologia "*design for verification*" num projeto de verificação.

SystemVerilog descendeu da linguagem Verilog, assim como podemos observar essa evolução na figura 2.2:

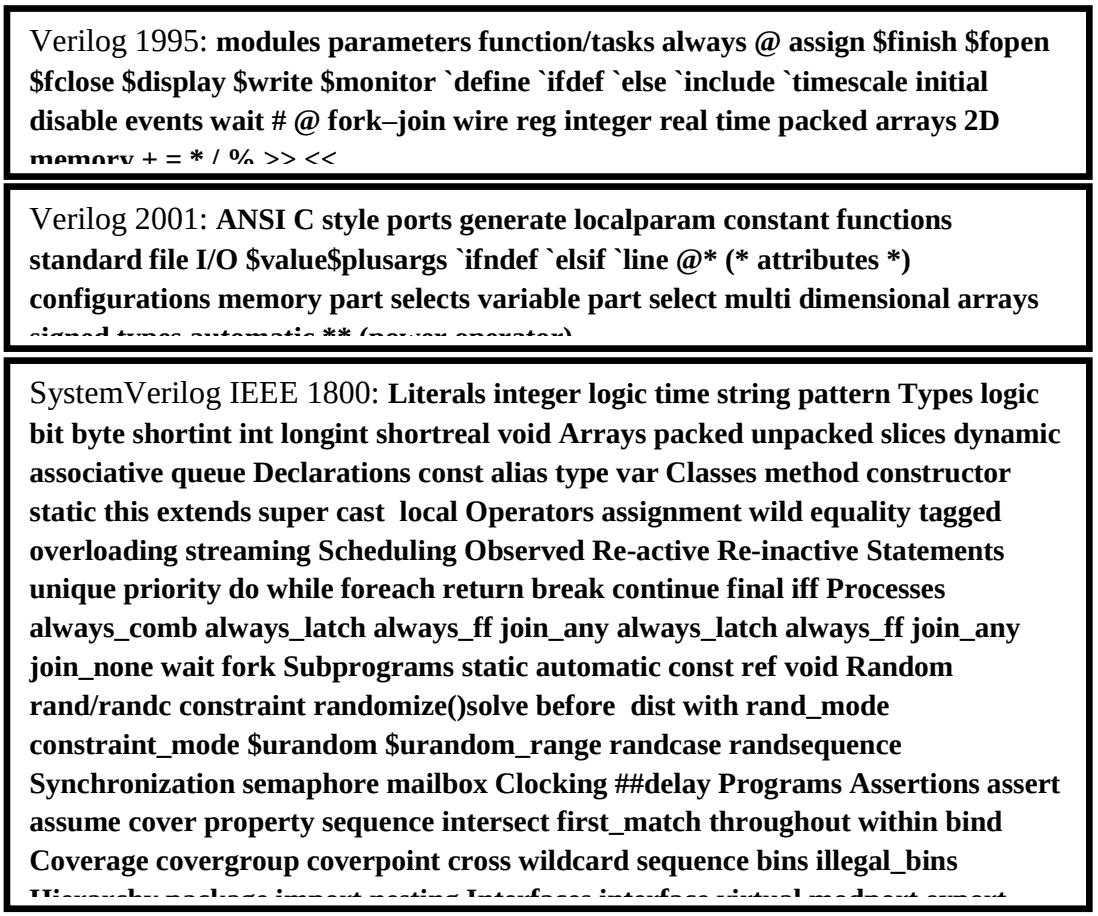


Figura 2.16. Expressões suportadas por SystemVerilog

As extensões SystemVerilog para Verilog descritas nesse capítulo destinam-se ao entendimento dessa Linguagem de Descrição de Hardware para a escrita de códigos sintetizáveis, sendo que o presente capítulo objetiva possibilitar o entendimento dessa parte da linguagem, bem como de sua utilização nos projetos de descrição de hardware.

Um código sintetizável é uma descrição desenvolvida para que uma ferramenta de síntese consiga decodificá-lo para as melhores portas lógicas possíveis. Sendo assim, nossa meta é desenvolver uma atividade para a descrição comportamental do hardware em SystemVerilog, e que essa descrição seja sintetizável.

Para um melhor entendimento será descrita toda a parte *front-end* do fluxo de desenvolvimento de um hardware. Iremos focar no *front-end*, por ser a etapa onde os engenheiros utilizam as linguagens de descrição de hardware para codificar a lógica do circuito.

2.2. Fluxo padrão de desenvolvimento de um projeto de hardware

Para desenvolver um projeto de hardware é necessário seguir um fluxo de desenvolvimento que contém algumas etapas de gerenciamento das atividades. A figura 2.3 mostra o fluxo padrão de desenvolvimento de um hardware:

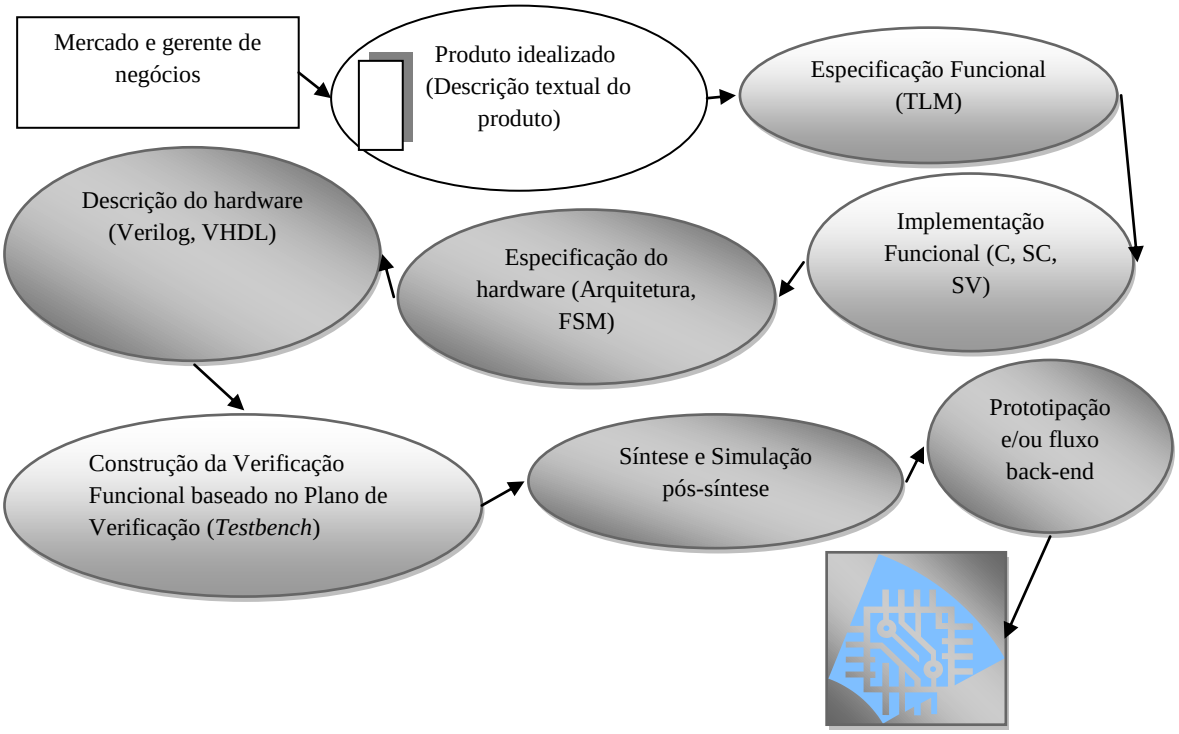


Figura 2.3. Fluxo padrão de desenvolvimento de um projeto de hardware

De acordo com a AMA(*American Marketing Association*), conceitua-se *marketing* como uma função organizacional e um conjunto de processos que envolvem a criação, a comunicação e a entrega de valor para os clientes, bem como a administração do relacionamento com estes, de modo que beneficie a organização e, consequentemente, seu público interessado. Algumas estratégias utilizadas nesse setor são: preço, pacote de produtos ou serviços, nicho do mercado, promoção, distribuição e gestão de marcas.

Após os grupos de negócios e *marketing* iniciarem o processo de criação do SOC, começa o processo de documentação dos requisitos e especificação do produto. A especificação deve ter um alto nível de abstração, ainda não havendo decisões em relação à implementação das funcionalidades, em termos da arquitetura-alvo a ser adotada, nem sobre os componentes de hardware ou software a serem selecionados. Conterá detalhes de alto nível, tais como funcionalidades a serem executadas, informações da freqüência e todos os requisitos requeridos pelo cliente. Caso o produto seja impossível de ser construído, é devolvido um documento detalhando todas as impossibilidades encontradas no processo de idealização, a fim de que a área de

negócios possa entrar em contato com o cliente e tomar as providências cabíveis. Caso contrário, é escrita uma especificação com todo o detalhamento do produto requisitado para que possa ser utilizado tanto pelo responsável pela especificação do hardware quanto para o responsável pela especificação da verificação funcional. É importante que qualquer mudança que ocorra dentro dessa especificação seja repassada para os responsáveis tanto pela especificação do hardware quanto para os responsáveis pela especificação da verificação funcional.

Uma vez construído o documento do produto, este é repassado para o setor de produtos e engenharia de sistema, que fica responsável em receber o documento do produto e gerar a especificação funcional. Este setor interage diretamente com o setor de marketing e com os engenheiros de implementação, tirando dúvidas e verificando problemas que podem tornar o projeto inviável. A equipe de produto e sistema é, pois, responsável por gerar a especificação funcional e o produto requisitado em alto nível, que incluirá as especificações do sistema e da verificação do hardware.

A implementação do sistema utiliza *Transaction level model* (TLM) para descrição no nível arquitetural de abstração, bem como para modelar os blocos identificados num estágio inicial de análise e exploração da arquitetura. Em geral, a modelagem em nível de transação capacita os engenheiros de sistema a especificarem comportamentos de blocos no alto nível de abstração, que tem por foco o comportamento dos blocos e a interação entre eles, sem haver preocupação com a sincronização existente no baixo nível.

Na próxima etapa do fluxo é desenvolvida a especificação do hardware. Neste momento, é crucial a descrição de alguns pontos detalhados para que os engenheiros de implementação não tenham nenhuma dúvida quanto ao produto que deve ser implementado. A seguir temos uma descrição dos pontos que devem fazer parte de uma especificação de hardware: visão geral do bloco, diagrama de blocos, interface de sinais, formas de ondas temporizadas, diagrama da máquina de estados do bloco, descrição da maquina de estado, registradores de controle do bloco (caso existam) e descrição dos possíveis caminhos críticos.

A especificação do hardware também deve incluir informações de síntese, tecnologia do processo que será utilizado, máxima frequência, energia e geometria (máxima área) utilizada para a síntese. Por tudo que já foi dito, fica claro que a especificação do hardware necessita ser realizada por alguém que conheça os mínimos detalhes da aplicação visada, e que qualquer mudança ocorrida dentro dessa especificação deve ser repassada para a equipe de verificação e documentada pelo gerente de projeto.

A implementação do hardware é um processo que utiliza uma linguagem de descrição de hardware em um nível mais baixo quando comparado ao nível de sistema. A descrição é feita em nível de sinais, utilizando um sinal de relógio para controlar e sincronizar os blocos implementados.

Posteriormente é iniciada a etapa de verificação funcional, cujo objetivo é certificar que a implementação inicial do RTL tem as mesmas características e funcionalidades do produto idealizado pelo cliente. Com vistas a tal objetivo, é criado um ambiente, denominado de ambiente de verificação, de modo a viabilizar uma convergência entre o RTL, o produto idealizado e a especificação funcional. O produto idealizado é construído numa linguagem de mais alto nível, na etapa de implementação do sistema, e irá servir para toda a etapa da verificação funcional como modelo de referência.

Na atividade de síntese é utilizada uma ferramenta que irá transformar o código RTL desenvolvido em postas lógicas. A partir de então, é feita a verificação desse

arquivo na etapa de simulação pós-síntese e o início do processo físico de desenvolvimento do hardware.

2.3. SystemVerilog para Descrição de Hardware

SystemVerilog foi desenvolvida para a utilização em qualquer etapa do fluxo de desenvolvimento de um *Hardware.* Todavia, nesse laboratório vamos aprender a desenvolver descrições que sejam sintetizáveis em qualquer simulador SystemVerilog. Para isso, deve-se observar quais expressões podem ser utilizadas quando estiver sendo desenvolvida a descrição.

2.3.1. Module

O modulo é a estrutura principal da modelagem de algum circuito. Com essa expressão, o engenheiro de descrição RTL irá criar um escopo para o circuito que será desenvolvido, ou parte dele. Após a descrição do módulo, deve-se finalizar a estrutura com uma outra expressão: *endmodule*. Essa expressão indica o final do módulo. Observe a figura 2.4.

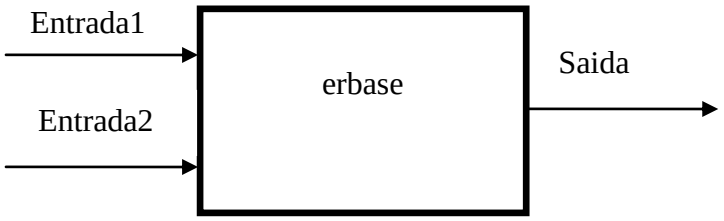


Figura 2.4. Exemplo de um módulo

A figura mostra um exemplo de módulo que podemos descrever em SystemVerilog. Apenas vendo esse bloco arquitetural, podemos definir que o nome do módulo será **erbase**, existindo duas entradas e uma saída de dados. Como não foi especificada a quantidade de bits que estarão entrando e nem saindo, por *default* SystemVerilog atribui que é um bit. Para criarmos os módulos precisaríamos da seguinte descrição:

```
module erbase (input Entrada1, input Entrada2, output Saida);

/*descreva a lógica do módulo */

endmodule
```

Figura 2.5. Descrição do módulo erbase

SystemVerilog é uma linguagem *case sensitive*, ou seja, diferencia letras maiúsculas de letras minúsculas.

2.3.2. Instanciação de módulos

A instanciação de módulos pode ser desenvolvida dentro de qualquer outro módulo. O conceito de descrição de módulos para serem instanciados torna SystemVerilog para design totalmente dirigido ao plano arquitetural. Cada módulo é descrito em separado, e no final da descrição de todos os módulos é desenvolvido um módulo, denominado *toplevel*, onde todos os módulos são instanciados. O código da figura 2.6 mostra como podemos instanciar um módulo dentro de outro:

nome_do_modulo #(conexão dos **parâmetros**) **nome_da_instância** (**conexão de entradas e saídas**);

```
multiplexador2_1 #(.WIDTH(.WIDTH)) mux1 (.sel(.sel_modulo_externo),
.entrada1(input_modulo_externo1),
.entrada1(input_modulo_externo2),
.saida(saida_modulo_externo));
```

Figura 2.6. Exemplo de instanciação de módulos

A sintaxe é descrita da seguinte forma: nome do módulo, parâmetros (caso exista), nome da instância, e ligações de todos os sinais dos módulos instanciados com as entradas e saídas dos módulos instanciados.

2.3.3. Tipos de Dados

A seguir serão mostrados os tipos de dados suportados por SystemVerilog. Serão definidos os dados que os simuladores SystemVerilog suportam para códigos sintetizáveis.

2.3.3.1 Nets

Os ***Nets*** foram herdados da linguagem de descrição de hardware Verilog. As principais características desses tipos de dados são:

1. Utilizados para conectar diferentes elementos;
2. Podem ser tratados como um fio físico;
3. Podem ser lidos ou atribuídos;
4. Nunca é guardado o valor;
5. Necessitam ser atribuídos sempre por uma porta do módulo.

O principal tipo de *net* utilizado numa descrição de hardware é o ***wire***. O elemento ***wire*** é definido como um simples fio que faz uma ligação direta entre dois pinos. As seguintes regras de sintaxe podem ser utilizadas para o ***wire***:

1. São utilizados para conectar portas ***inputs*** e ***outputs*** de instâncias de módulos com algum outro elemento do projeto;

2. São utilizados como ***inputs*** e ***outputs*** dentro de um módulo;
3. ***Wires*** devem ser conectados em algum pino, e não podem guardar o valor sem serem conectados.
4. Não podem ser utilizados do lado esquerdo de um = e <= dentro de um **always;**
5. São aceitos do lado esquerdo apenas quando utilizamos o ***contínuos assign***;
6. Podem ser utilizados apenas para modelagem combinacional.

O código da figura 2.7 mostra um exemplo do uso do elemento ***wire***:

```
wire     A, B, C, D, E; //fios de apenas um bit
wire    [8:0] array_de_fios; // conjunto de 9 fios criados
reg i;
assign     A = B  &  C; // usando wire com bloco assign
always @( B  or  C )  begin
        I   =   D | E;        // usando wire dentro de um bloco always
end

meu_modulo X1 (.in(D),  // utilizando wire como input


.out(A) ); // utilizando wire como input
```

Figura 2.7. Exemplo do uso de wire

2.3.3.2 reg

O uso do elemento ***reg*** é similar ao uso do ***wire***, podendo, porém, ser utilizado para guardar informações semelhantes a um registrador. São utilizados para desenvolver uma lógica lógica combinacional ou sequencial. Algumas regras de sintaxe devem ser observadas:

1. Elementos ***reg*** podem ser conectados a portas de entrada de uma instância de módulo. Note que ***reg*** não pode se conectar com uma porta de saída de uma instância de módulo.
2. ***reg*** podem ser utilizados como ***outputs*** dentro de uma declaração de módulo, porém não podem ser utilizados como ***inputs*** do módulo desenvolvido.
3. ***reg*** só é utilizado do lado esquerdo de uma expressão, dentro de um bloco procedural ***always@*** com os sinais de atribuições = ou <=.
4. ***reg*** só é utilizado dentro de um ***initial*** com o sinal =.
5. ***reg*** não pode ser utilizado com blocos ***assign***.
6. ***reg*** pode ser utilizado para criar registradores quando utilizado com algum evento do sinal principal gerenciador do circuito. Geralmente são utilizados com o CLOCK.

7. ***reg*** pode ser utilizado para criar lógicas de circuitos combinacionais e sequenciais.

O exemplo da figura 2.8 mostra o uso correto do elemento ***reg***.

```
wire     a, b;
reg      i, j, k;  // reg simples
reg    [8:0] array_de_regs; // conjunto de 9 fios criados
always @(  a | b) begin
        i  =  A | B; // utilização de um reg recebendo dados de um wire
end
initial begin //utilizando reg num bloco initial. Initial não é sintetizável
        J = 1'b1;
        #1;
        J = 1'b0;
end
always @(posedge clock) begin
        K <= I; // usando reg para criar um registrador com gatilho na borda de
subida
end
```

Figura 2.8. Exemplo do uso do reg

2.3.3.3 logic

O elemento ***logic*** foi inserido por SystemVerilog para eliminar o problema de decisão do uso de **reg** ou **wire.** O objetivo principal da inserção do elemento **logic** é deixar que a escolha entre **reg** e **wire** seja feita pelo sintetizador, definindo quais sinais na síntese desenvolvida.

O elemento **logic** não permite o uso de múltiplos valores. Os elementos **wire** e **reg** podem ter quatro comportamentos: 1, 0, X ou Z. Um e zero são os valores possíveis de um bit, verdadeiro ou falso. Nos sistemas digitais os valores 0 ou 1 representam um nível de tensão. O valor 0 representa um nível de tensão de 0V até 0,8V, enquanto o valor 1 representa um valor de tensão entre 3.3V e 5V. Caso o valor da tensão do sinal esteja entre 0,8V e 2V, no intervalo aberto, o sinal é dado como indeterminado. Esse problema pode ser causado por ruídos ou qualquer outro tipo de resistência que afeta o sinal. Em Verilog, essa indeterminação é representada pelo valor X. O valor Z quer dizer que o sinal está em alta impedância, sendo impedância a oposição que um circuito faz à passagem da corrente elétrica. Pode-se dizer que é a "resistência" dos circuitos de corrente alternada. Podemos observar na figura 2.9 como foi padronizado os valores 0, 1 ou X.

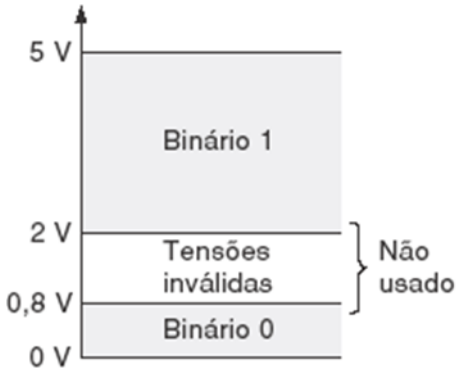


Figura 2.9. Valores de tensão utilizados para representação de sistemas digitais

Portanto, com o **logic** não temos a possibilidade desses quatro tipos de representação, apenas dois: 0 e 1. Isso facilita a distinção do sintetizador e de todos os sinais que podem ser sintetizados com valores determinados. O valor Z pode ser visto na saída de um tri-state. O **logic** pode ser definido como um **reg** em um local e um **wire** em outra parte do código, como se pode observar na figura 2.10.

```
assign a = b ^ c; // wire style
always (c or d) a = c + d; // reg style
myModule module(.out(a), .in(xyz)); // wire style
```

Figura 2.10. Exemplo do uso do *logic* como *reg* e *wire* num mesmo código

2.3.3.4 typedef

SystemVerilog definiu uma forma de criação de tipos utilizando a palavra-chave **typedef.** Esse formato de criação de tipos foi baseado no formato da linguagem C, sendo utilizado para criação de novas variáveis que terão o mesmo tipo de dado declarado anteriormente. O **typedef** é utilizado para definição de tipos complexos de dados. O exemplo da figura 2.11 mostra seu uso.

```
typedef int unsigned uint;
...
uint a, b; // Duas variáveis do tipo uint
```

```
typedef reg [7:0] reg_8;
reg_8 byte;
```

Figura 2.11. Exemplo da estrutura de um typedef

2.3.3.4.1 Definição local typedef

A criação de tipos pode ser definida localmente, dentro de um **package**, ou externamente, num escopo de unidade de compilação. Quando é definido um tipo, ele será utilizado dentro da parte do projeto que foi definido. A figura 2.12 mostra o uso do **typedef** dentro de um módulo.

```
module alu (...);
typedef logic [3:0] meu_reg;
meu_reg   opA, opB; // Variáveis do tipo meu_reg
meu_reg [7:0] data; // Um registrados de 32 bits utilizando 8 vetores meu_reg
...
endmodule
```

Figura 2.12. Exemplo de uso do typedef local

2.3.3.4.2 Definição compartilhada do typedef

Quando é necessário utilizar o tipo em vários módulos diferentes, o **typedef** pode ser declarado num ***package*** (estrutura de SystemVerilog). Essa definição pode ser referenciada diretamente, ou importada dentro de cada módulo que fará o uso. O exemplo ilustrado na figura 2.13 mostra o uso do **typedef** dentro de um **package.**

```
package chip_types;
                                    `ifdef TWO_STATE
        typedef bit mytype_t;
    `else
        typedef logic mytype_t;
    `endif
endpackage
module counter
(output chip_types::mytype_t [15:0] count,
input chip_types::mytype_t clock, resetN);
always @(posedge clock, negedge resetN)
if (!resetN) count <= 0;
else count <= count + 1;
```

Figura 2.13. Exemplo do uso do typedef compartilhado

2.3.3.5 enum

O tipo **enum** tem o objetivo de fornecer meios para declaração de uma variável abstrata onde se pode descrever uma lista específica de valores. Cada valor é identificado com um nome definido pelo usuário. Esse nome também é denominado de *label*. A figura 2.14 mostra o uso do **enum**.

```
enum {circulo, elipse, quadrado, retangulo} elementos_geometricos;
enum {red, green, blue} padrão_rgb;
```

Figura 2.14. Exemplo do uso da estrutura enum

Em Verilog não existe esse tipo de dado. Para criar pseudo *labels* de valores é necessário definir parâmetros constantes para representar cada valor, além de definir valores a cada constante. Alternativamente, pode-se utilizar o **`define** para desenvolver nomes macros com valores específicos para cada nome**.** O exemplo da figura 2.15 mostra uma comparação de como desenvolver a mesma estrutura com **parameter** e com **enum.**

```
parameter       WAIT = 0,
                LOAD = 1,
                STORE = 2;
reg [1:0] State, NextState;

enum {WAIT, LOAD, STORE} State, NextState;
```

Figura 2.15. Exemplo mostrando a criação de estados em Verilog e em SystemVerilog

Tal tipo de dado também pode ser utilizado em conjunto com o **typedef.** Desta forma, podemos criar estruturas mais complexas, como máquinas de estados finitos de diversos tamanhos. No exemplo da figura 2.16 temos uma comparação entre o desenvolvimento de uma máquina de estados finito em Verilog utilizando a estrutura **`define** e em SystemVerilog com **typedef enum**.

Verilog

```
`define FETCH 3'h0
`define WRITE 3'h1
`define ADD 3'h2
`define SUB 3'h3
`define MULT 3'h4
`define DIV 3'h5
`define SHIFT 3'h6
`define NOP 3'h7
module controller (output reg read, write,
                                    input wire [2:0] instruction, // pode ser
FECTH, WRITE…
                   input wire clock, resetN);
```

SystemVerilog

```
package chip_types;
        typedef enum {FETCH, WRITE, ADD, SUB,
                      MULT, DIV, SHIFT, NOP } instr_t;
endpackage
import chip_types::*; // import package definitions
module controller (output logic read, write,
                   input instr_t instruction, // pode ser FECTH, WRITE…
                   input wire clock, resetN);
```

Figura 2.16. Comparação entre uma máquina de estados em Verilog e SystemVerilog

2.3.4 Operadores

Assim como qualquer outra linguagem, SystemVerilog tem um conjunto de operadores que dão o suporte necessário à linguagem. Herdou todos os operadores existentes na linguagem Verilog, com acréscimo de alguns.

2.3.4.1 Operadores Lógicos

Os operadores lógicos existentes podem ser divididos em duas tabelas: a dos operadores unários e a dos operadores binários. Aqueles são responsáveis por executar operações com um único operando, enquanto estes utilizam dois operandos. Um exemplo de seu funcionamento é a operação AND entre dois bytes: a saída dessa expressão é um número gerado por uma AND de cada bit dos bytes.

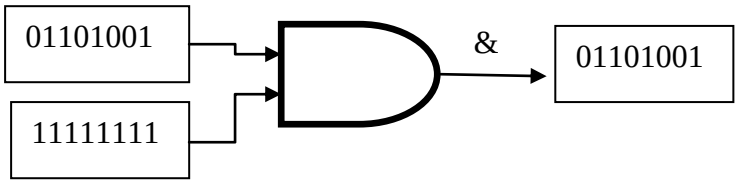


Figura 2.17. Exemplo do uso de um operador unário

Os operadores unários da linguagem SystemVerilog podem ser vistos na tabela 2.1.

Tabela 2.1. Tabela de operadores unários

Operador	Exemplo	Descrição
~	C = ~D	Operador de negação. Se D =1 então C=0;
!	C = !D	Operador NOT lógico;
<<		Translação (bit a bit) à esquerda;
>>		Translação (bit a bit) à direita com sinal;
>>>		Translação (bit a bit) à direita sem sinal. O bit de sinal será 0;
[] () (*tipo*)	(integer)	Máxima precedência: separador, indexação, parâmetros, conversão de tipo;

Além dos operadores lógicos unários, também temos os operadores lógicos binários. Podemos observar todos os operadores lógicos binários na tabela 2.2.

Tabela 2.2. Operadores Lógicos Binários

Operador	Exemplo	Descrição
<, <=, >=, <	C = A >= B;	Utilizado para fazer comparações entre 2 operandos;
==, !=	if(c==a)...	Igualdade: igual e diferença entre operandos;
&	C = A & B	AND bit a bit;
^	C = A ^ B	XOR bit a bit;
|	C = A | B	OR bit a bit;
&&	if(A && B)	Operador lógico **e** condicional;
||	C = A & B	Operador lógico **ou** condicional;
=, +=, -=	C += D	Atribuição;

2.3.4.2 Operadores Aritméticos

Os operadores aritméticos existentes na linguagem SystemVerilog podem ser vistos na tabela 2.3.

Tabela 2.3. Operadores Aritméticos

Operador	Exemplo	Descrição
<, <=, >=, <	C = A >= B;	Utilizado para fazer comparações entre 2 operandos;
==, !=	if(c==a)...	Igualdade: igual e diferença

		entre operandos;
&	C = A & B	AND bit a bit;
^	C = A ^ B	XOR bit a bit;
|	C = A | B	OR bit a bit;
&&	if(A && B)	Operador lógico **e** condicional;
||	C = A & B	Operador lógico **ou** condicional;
=, +=, -=	C += D	Atribuição;

2.3.4.3 Operador Condicional

Tabela 2.4. Operador Condicional

Operador	Exemplo	Descrição
?:	mux_aux = sel ? a : b;	Condicional: **if-then-else** compacto;

O operador condicional expressa de forma curta uma expressão condicional IF. No exemplo da tabela 2.4 a condicional seria semelhante a:

```
if (sel)
        assign mux_out = a;
else
     assign mux_out=b;
```

2.3.4.3.1 Case, Unique Case, Priority case

A linguagem de descrição de Hardware SystemVerilog definiu métodos que diferenciam as três formas da expressão case: **case**, **unique case** e **priority case**. Em **case** todas as condicionais são desenvolvidas na sequência que aparece na declaração dos casos. Podemos observar um exemplo na figura 2.18.

```
case (prior)
        0: op = a;
        0,1,2 : op = b;
        3: op = c;
endcase
```

Figura 2.18. Exemplo case

No exemplo, existem duas condições para que o **0** seja executado. Como temos um **case,** serão executadas em sequência as duas primeiras condições; a segunda condicional **0** irá sobrescrever a primeira e a execução terá a saída **b**. Essa escolha depende do compilador que fará a síntese.

Quando utilizamos o **unique case** o compilador afirma que não há nenhum caso de itens que se sobrepõem e, portanto, que ele é seguro para o caso de itens a serem avaliados em paralelo. O exemplo da figura 2.19 demonstra o uso do **unique case**.

```
unique case (fullc)
        0,1: op = a;
        1,2: op = b;
        3 : op = c;
endcase
```

Figura 2.19. Exemplo unique case

O exemplo da figura 2.19 mostra que temos um **unique case,** com duas condições iguais. Nesse caso, o simulador/compilador irá mostrar um *Warning*, chamando atenção do projetista para que corrija o **unique case**.

No uso do **priority case** é mostrado um *warnig,* que identifica duas condições iguais e apenas executará o primeiro caso, ou seja, o caso prioridade.

2.3.4.3.2 If, Unique if, Priority IF

SystemVerilog também proveu algumas mudanças com a expressão condicional **if**. Existem três tipos de expressões if: **if**, **unique if, priority if**. Em Verilog um **if** é avaliado como um booleano, de modo que se o resultado da expressão é 0 ou X, o teste é considerado falso. SystemVerilog adiciona as palavras-chave **unique** e **priority**, que podem ser utilizadas antes de um **if**. Caso uma dessas palavras-chaves seja utilizada, poderá gerar um erro em tempo de execução, se nenhuma condição corresponder à verdade, a não ser que haja uma condição **else** explícita.

Utiliza-se **priority if** para fazer a análise das condições em sequência, e a resposta será a primeira condição verdadeira encontrada.

Já **unique if** é utilizada para lógica paralela, de modo que as condições são analisadas paralelamente. Apenas uma condição deve ser verdadeira.

2.3.4.4 Modelagem Comportamental

A modelagem comportamental permite um nível de abstração maior na descrição de um hardware. A partir da descrição do hardware em uma linguagem HDL, podemos verificar se a lógica do circuito está correta e se poderá ser sintetizável. Para a descrição comportamental de um código sintetizável em Verilog, temos apenas dois tipos de blocos para utilizar: o continuous assignment (assign) e o bloco procedural **always.** Em SystemVerilog foram inseridos outros alguns blocos procedurais: **always_ff**, **always_comb**, **always_latch.**

2.3.4.4.1 Continuous assignment (assign)

O ***continuous assignment*** é utilizado apenas para a descrição da lógica combinacional. Qualquer mudança no lado direito de uma expressão causa atualização imediata do lado esquerdo. Como descreve apenas lógica combinacional, pode apenas descrever circuitos combinacionais. Na figura 1.19 pode-se observar o uso do ***assign*** para a descrição de um circuito combinacional em que cinco saídas diferentes recebem uma atualização dos valores de **y1, y2, y3, y4** e **y5,** imediatamente após executar a expressão. As cinco

expressões serão realizadas quando houver alteração nos valores de **a** e **b** no código. A execução de todos os **assign's** ocorre paralelamente. Observe o exemplo da figura 2.20.

```
module gates (input [3:0] a, b,
output [3:0] y1, y2, y3, y4, y5);
/* 5 diferentes portas de 2 entradas
 e 4 bits */
        assign y1 = a & b; //AND
        assign y2 = a | b; //OR
        assign y3 = a ^ b; //XOR
        assign y4 = ~( a & b ); //NAND
        assign y5 = ~( a | b ); //NOR
endmodule
```

Figura 2.20. Exemplo do uso do assign

2.3.4.4.2 Blocos Procedurais

Os blocos procedurais são seções contendo atribuições que são executadas linha por linha, semelhante a um software convencional. Todavia, podem-se utilizar vários blocos procedurais para interagirem concorrentemente.

2.3.4.4.2.1 Always

O bloco procedural **always** foi herdado da linguagem de descrição de hardware Verilog. Utilizando apenas esse bloco, podemos descrever lógicas combinacionais, latches e lógicas sequenciais. O **always** sempre executa um evento ou uma lista de eventos que existe no cabeçalho do bloco. Essa lista tem que ser escrita entre parênteses e após o **@.** Só é executado o bloco procedural quando todas as condições do evento forem satisfeitas. O evento poderá ocorrer na borda de subida de um sinal ou na borda de descida (**posedge**, **negedge**), e geralmente esse detalhamento é observado em circuitos sequenciais. Observe no exemplo da figura 2.21 como é desenvolvido um **always** para um circuito sequencial e para um circuito combinacional. Numa descrição de hardware, quando temos uma lista de eventos com o uso do (*), significa que aquele bloco procedural é sensível a qualquer modificação dos sinais do módulo.

```
//Circuito sequencial
always @(posegde clk or negedge reset) begin
end
//Circuito combinacional
always @(*) begin
end
```

Figura 2.21. Exemplo do uso do bloco procedural always

2.3.4.4.2.2 always_ff

SystemVerilog inseriu o bloco procedural **always_ff** para auxiliar o engenheiro de descrição a definir uma lógica sequencial. Com esse bloco é possível apenas descrever lógicas sequenciais. Dessa forma o engenheiro detalha para todos da equipe que o objetivo daquele bloco é desenvolver uma lógica sequencial. O exemplo da figura 2.21 mostra um exemplo do uso do **always_ff.**

2.3.4.4.2.3 always_comb

O **always_comb** é utilizado para a descrição de blocos combinacionais. Com a finalidade de tornar a leitura da descrição comportamental do módulo mais simples, SystemVerilog define essa palavra para deixar claro que a lógica descrita do bloco será combinacional. Na figura 2.21 pode-se observar um exemplo do uso do **always_comb**.

2.3.4.4.2.4 always_latch

Raras vezes há a necessidade da criação de um ***latch*** dentro de um circuito. Seu uso é restrito, devido aos problemas de temporização que pode acarretar para o chip como um todo. Para definir a criação de um ***latch*** em SystemVerilog, deve-se utilizar o bloco **always_latch**. A figura 2.22 mostra um exemplo de como se pode utilizar o **always_latch.**

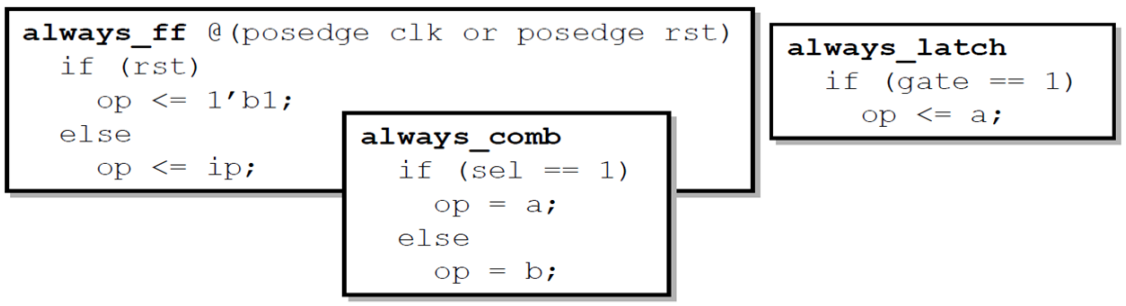


Figura 2.22. Exemplo do uso do always_ff, always_comb, always_latch

2.3.5 Blocking (=) X non-blocking (<=)

A linguagem Verilog tem dois operadores de atribuição: **blocking (=)** e **non-blocking (<=).** A atribuição de **blocking** é representada com um único sinal de igual (=), e a atribuição **non-blocking** é representada com um ***token*** de menor junto com um igual (<=).

```
out = in; // blocking assignment
out <= in; // nonblocking assignment
```

N**on-blockinng** ("<=") é o operador de atribuição de escolha quando se quer especificar o comportamento de um sinal na borda sensível de um circuito.

Afirma que todos os sinais farão a transferência de dados em todo o sistema, de acordo com toda a lista de eventos especificados, ocorrendo apenas na borda de subida ou na borda de descida do sinal da lista de sensibilidade. Todas as modificações dos sinais ocorrem simultaneamente.

Embora as descrições através do uso do **blocking** sintetizarem adequadamente, eles não podem simular esse tipo de comportamento, visto que a atribuição de valores utilizando o **blocking** irá atribuir os valores imediatamente após a leitura da linha, sendo que esse comportamento poderia trazer sérias consequências para a construção de circuitos sequenciais. Sempre que for necessário o uso de qualquer atribuição, opte pelo uso do **non-blocking**; porém, existem casos em que é necessário o uso do blocking, a exemplo do desenvolvimento de uma máquina de estados finitos. Devemos saber o próximo estado antes do final da borda do clock para que possamos repassar essa informação no circuito sequencial que guarda os valores dos estados. Caso isso não ocorra, a máquina se comportará de forma incorreta. Observe o exemplo 2.23, onde temos dois códigos semelhantes com diferentes comportamentos.

```
A=1;
B = 2;
C = 3;
always_comb  begin
…
A=B;
C = A +B;
```

```
A=1;
B = 2;
C = 3;
always_comb  begin
…
A<=B;
C <= A +B;
```

Figura 2.23. Exemplo do uso do blocking e non-blocking

A figura 2.23 mostra a diferença de comportamento quando se utiliza **blocking** e **non-blocking**. No primeiro quadro, temos a seguinte interpretação: as variáveis **A**, **B** e **C** são iniciadas com 1, .2 e 3 respectivamente. Dentro do bloco procedural **always_comb** é lida a primeira linha e feita a atribuição. Portanto **A** será igual a 2, imediatamente. Quando for executada a próxima linha será atribuído a **C** o valor de 2 + 2, ou seja, **C** será igual ao valor 4.

No segundo quadro temos a seguinte interpretação: as variáveis **A**, **B** e **C** são iniciadas com 1, .2 e 3, respectivamente. Dentro do bloco procedural **always_comb** é lida a primeira linha, depois a segunda e a atribuição é feita apenas na mudança de borda do sinal de gerenciamento simultaneamente em todos os sinais. Portanto **A** será igual a 1 até a borda e **C** terá o valor de 1 + 2, ou seja, **C** será igual ao valor 3.

2.4 Instalação do ambiente de desenvolvimento QUARTUS II

O software Quartus II, da ALTERA Corporation, é uma IDE (Integrated Development Enviroment) CAD (Computer-Aid Design) voltada para o design de hardware digital, permitindo a realização de todas as etapas envolvidas no projeto de um sistema digital, como a criação e utilização de esquemáticos, linguagens de descrição de hardware

(VHDL, Verilog...), simulação de circuitos (funcionalidade e simulação de *timing*) e síntese de circuitos para FPGA's, PLD's.

O Quartus II trabalha com um sistema orientado a projetos, ou seja, o software associa diversos arquivos a um projeto, e possibilita a utilização de componentes básicos, simplificando a elaboração de projetos mais complexos, uma vez que as funcionalidades de um grande sistema podem ser divididas em diversos sistemas mais simples e por fim, agrupados.

O software permite que sejam utilizadas uma interface gráfica e uma interface de linha de comando, deixando a critério do usuário fazer todo o processo em uma ou outra interface, ou mesmo com alternância de fases.

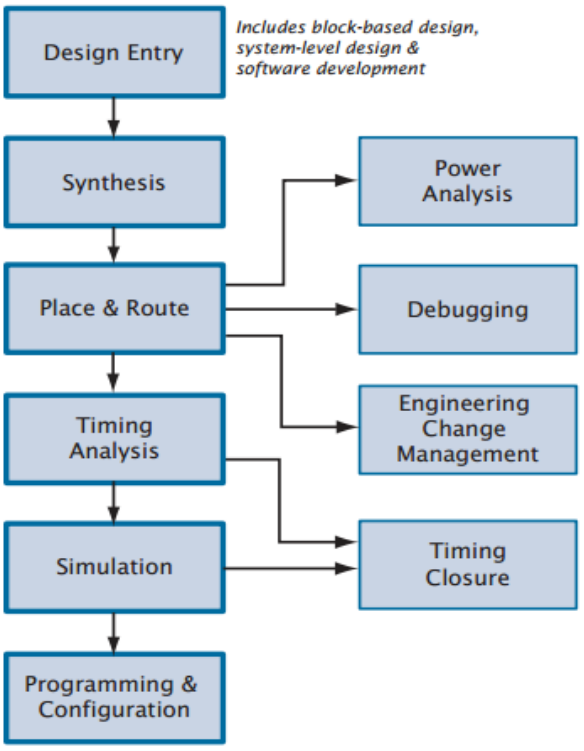


Figura 2.24. Fluxo de Design do Quartus II

A IDE inclui um compilador modular, que por sua vez inclui os seguintes módulos (módulos marcados com * são opcionais durante a compilação, dependendo das suas configurações):

■ Analysis & Synthesis
■ Partition Merge*
■ Fitter
■ Assembler*
■ TimeQuest Timing Analyzer*
■ Design Assistant*
■ EDA Netlist Writer*
■ HardCopy®
■ Netlist Writer*

2.4.1 Como adquirir o Quartus II FREE

1. Acesse o site da Altera: http://www.altera.com;
2. Clique no link Download Center;

Figura 2.25. Tela inicial do site da altera

3. Na caixa Download Individual Components, Step 1, clique em **Quartus II Web Edition**, em Step 2 clique na mais nova versão (11.1 Service Pack 2) e depois clique em **GO**;

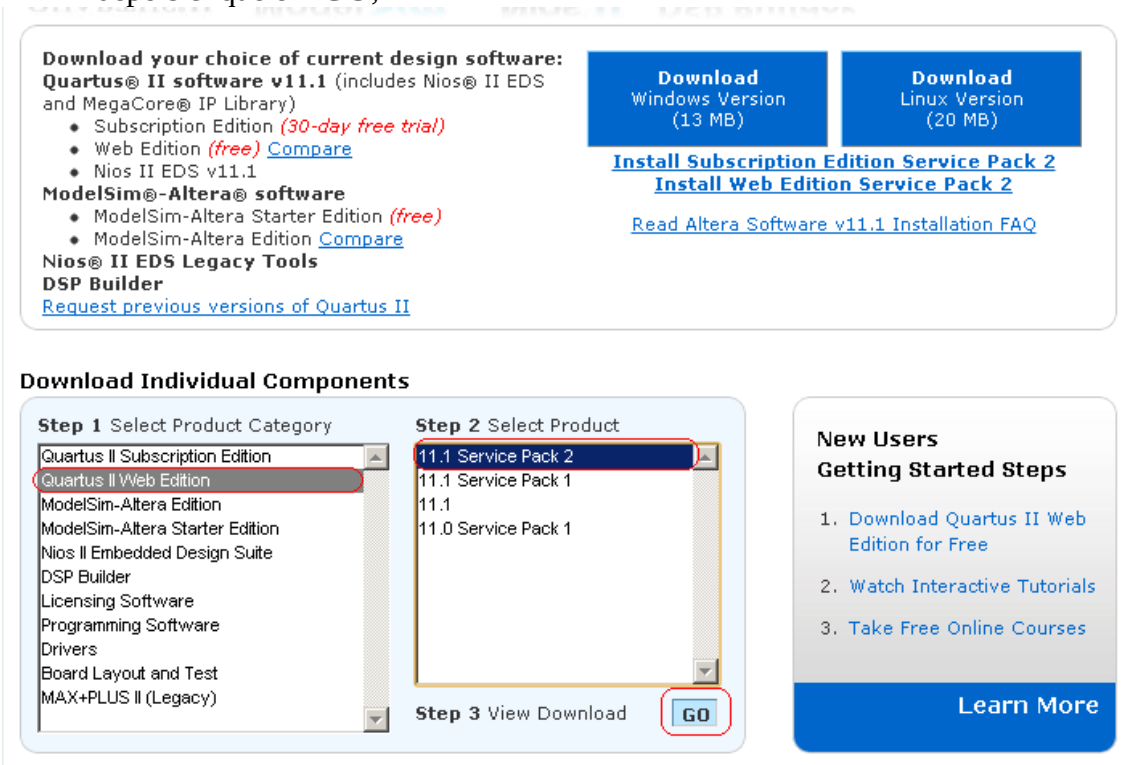


Figura 2.26. Página para fazer o download do QuartusII

4. Escolha a versão correspondente ao seu Sistema Operacional (Windows ou Linux);
5. Faça seu cadastro FREE no site e clique em **Continue**.

1. **Instalação do Quartus II (Windows)**

Ao abrir o arquivo executável, escolha a pasta para onde deseja extrair os arquivos de instalação (**Destination folder**) clicando em **Browser**, e então clique em **Install**.

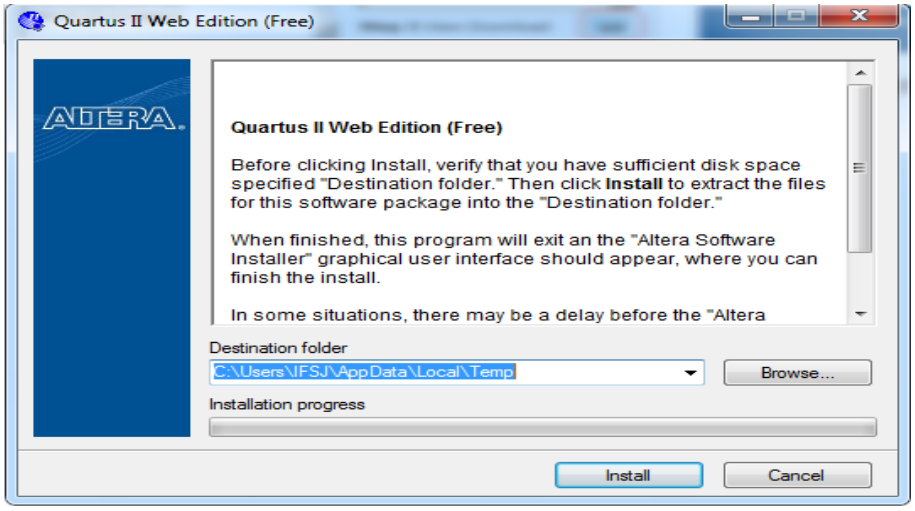


Figura 2.27. Tela inicial do processo de instalação do Quartus II

Feito isso, os arquivos de instalação serão extraídos para o seu computador, mas a instalação de fato ainda não ocorreu. Neste momento, será exibida uma imagem; clicando apenas em **Next,** aparecerá a imagem abaixo, onde o usuário deve clicar em "**I agree to the terms of the license agreement"** para aceitar os termos de licença do produto.

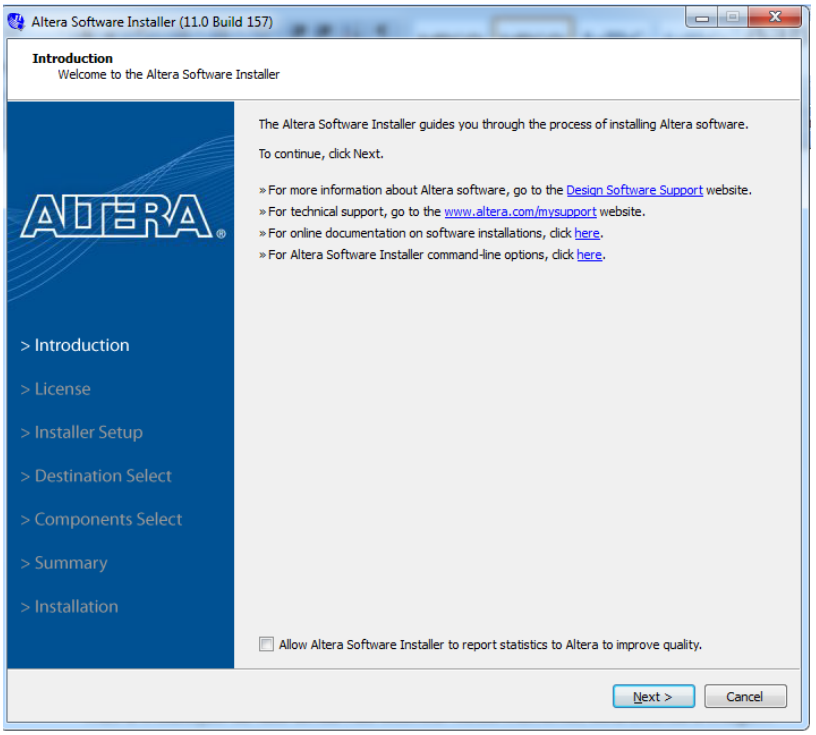


Figura 2.28. Tela secundária do processo de intalação

Na tela seguinte, deve ser escolhida a pasta destino (clicando em **Browser**), onde o software será instalado. Clique em **Next** e será exibida a tela abaixo. Nesse momento, são exibidos componentes que são instalados com o Quartus II, ficando a critério do usuário decidir quais destes instalar ou não.

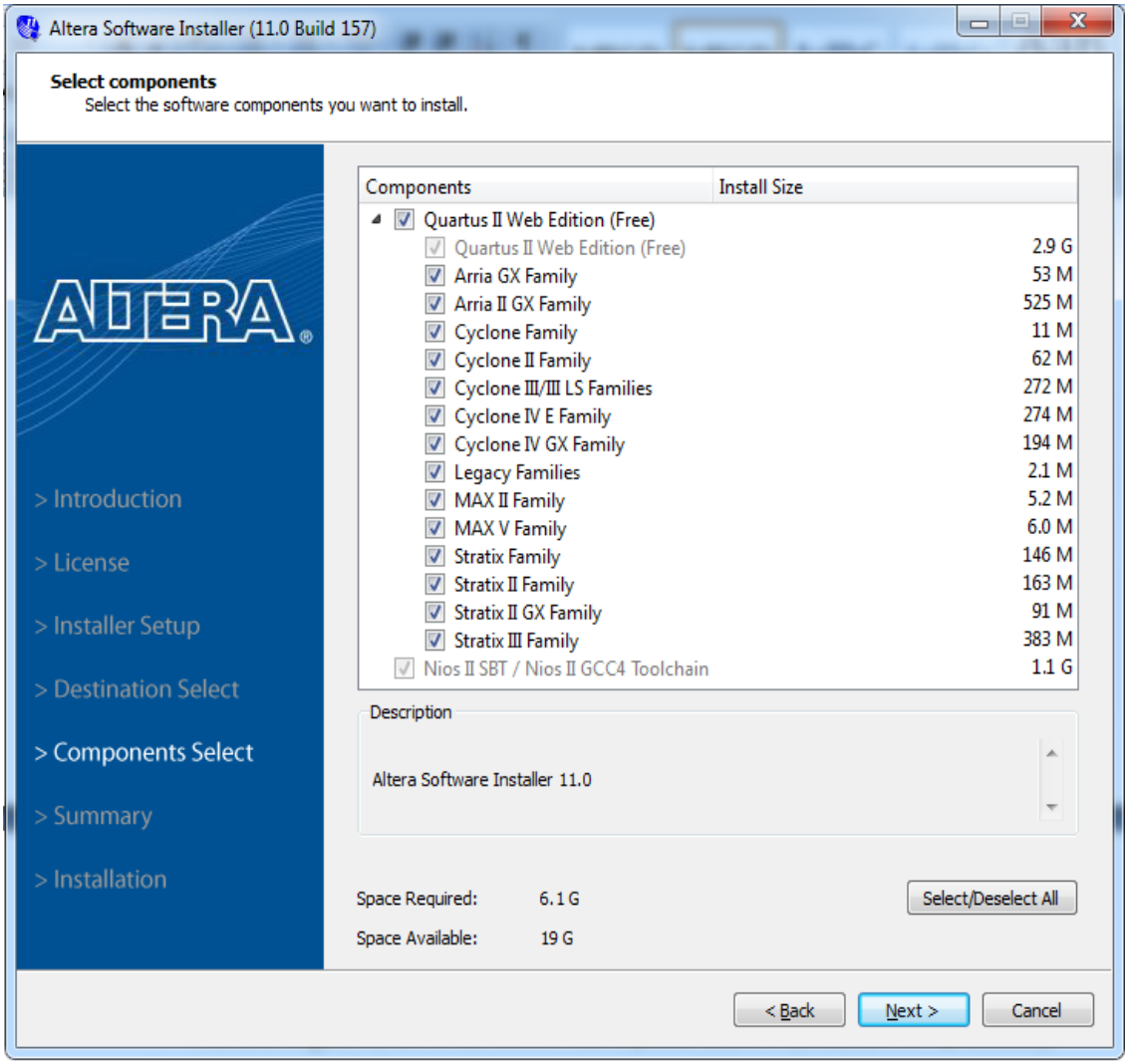


Figura 2.29. Tela de escolha dos componentes FPGAs que serão intalados

Clicando em **Next** e em **Finish,** nas telas seguintes, o software começa a ser instalado de fato. O processo pode ser relativamente demorado, por se tratar de uma ferramenta de grande porte.

2. Como utilizar no Windows

Para que seja utilizado no Windows, após ser feita a instalação é necessário apenas que o software seja aberto com o duplo clique no respectivo ícone.

3. Primeira tela

A primeira tela que é apresentada (ver figura abaixo) oferece opções importantes, quais sejam: **Create a New Project** e **Open Existing Project**, que indicam a criação de um projeto novo e a abertura de um projeto já existente no PC, respectivamente.

Figura 2.30. Tela inicial do Quartus II

Fechando essa tela inicial, é exibida a tela principal do Quartus, que normalmente é o ambiente de trabalho dos desenvolvedores. Na figura abaixo são numeradas as áreas que compõem este ambiente:

1. Navegador do Projeto – exibe os arquivos presentes no projeto;
2. Tarefas – mais utilizado para analisar como anda a compilação do projeto;
3. Mensagens – exibe as mensagens de execução e de compilação (como erros e sucessos, por exemplo);
4. Tela principal – exibe o código escrito.

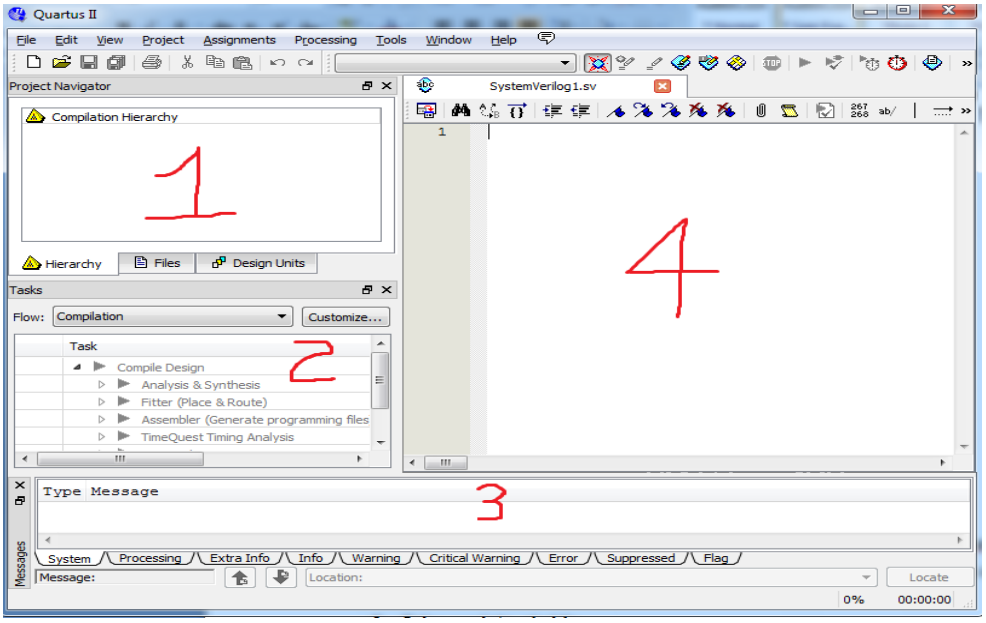


Figura 2.31. Tela de desenvolvimento do Quartus II

4. Criar projeto rápido

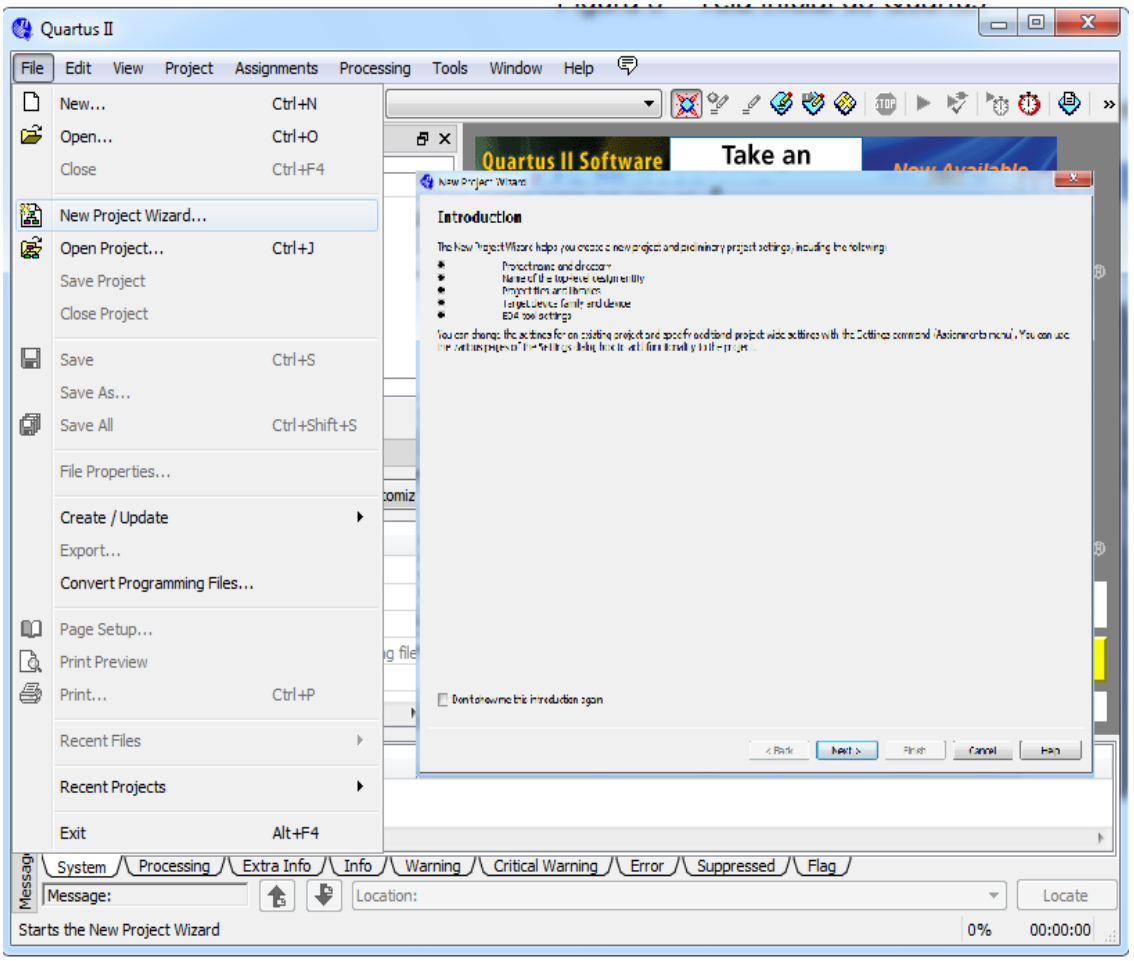


Figura 2.32. Primeira tela para a criação de um projeto

Inicie o software e utilize o menu **File > New Project Wizard** (figura acima)**.** Caso apareça uma tela de introdução, clique em **Next**. Será necessário especificar o caminho do seu projeto e, no nome do projeto, é recomendada a utilização do mesmo nome da entidade top-level do seu design systemVerilog, como por exemplo *tutorial*. Nas próximas telas selecione **Next** (haja vista que não vamos sintetizar o circuito numa FPGA; apenas faremos simulações dos nossos designs dentro do ambiente Quartus). Perceba que foi alterada a área do Navegador do Projeto.

Criado o projeto, clique em **File >New** e em seguida escolha **SystemVerilog HDL File**. Se a intenção for de projetar um circuito (ao invés de escrever código), a opção deverá ser de **Block Diagram/Schematic File**; neste caso, surgirão botões para funções voltadas à montagem do circuito, como adicionar componentes, fios, etc.

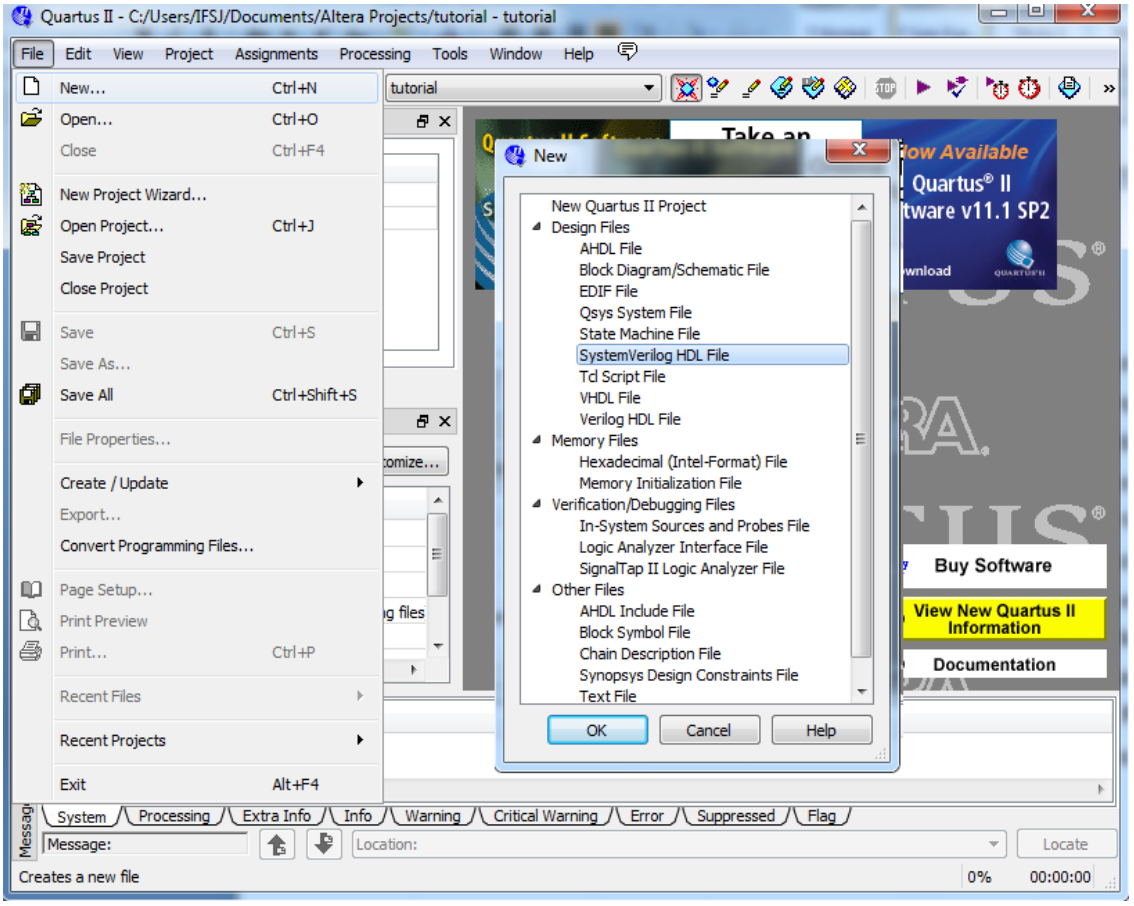


Figura 2.33. Menu para criar o arquivo SystemVerilog

5. **Selecionando o FPGA**

Para selecionar o FPGA que será utilizado no projeto, clique em **Assignments** > **Device** e surgirá a seguinte tela:

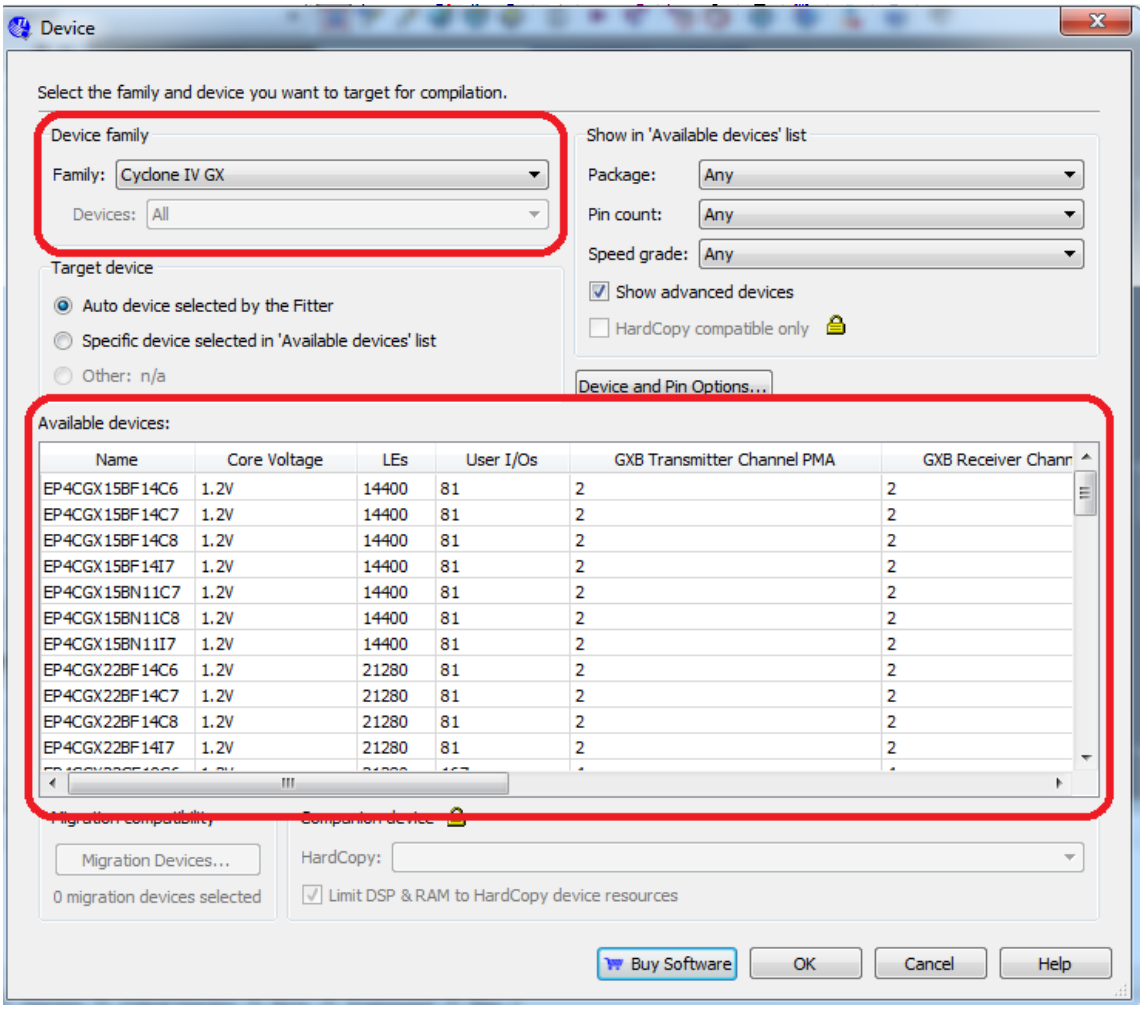


Figura 2.34 Tela para a escolha do tipo de FPGA utilizado

Em **Device family** pode-se selecionar a família do FPGA desejado, enquanto em **Available devices** são exibidos os dispositivos disponíveis da família escolhida, com seus respectivos nomes, tensão, tamanho de memória, entre outras características. Clique em **OK** para confirmar.

Para verificar qual o FPGA que está sendo usado no projeto, basta observar a região **Project Navigator: Entity**, normalmente localizado à esquerda do ambiente do Quartus II:

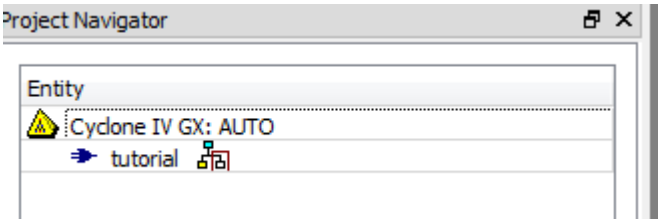


Figura 2.315. Aba que mostra o projeto criado

6. Onde Compilar

Uma vez terminado o programa, devemos compilá-lo para, em seguida, fazer a simulação. A compilação é feita clicando na aba **Processing > Start Compilation**, ou clicando no botão referente a esta tarefa na barra de ferramentas do software, conforme indicado na figura:

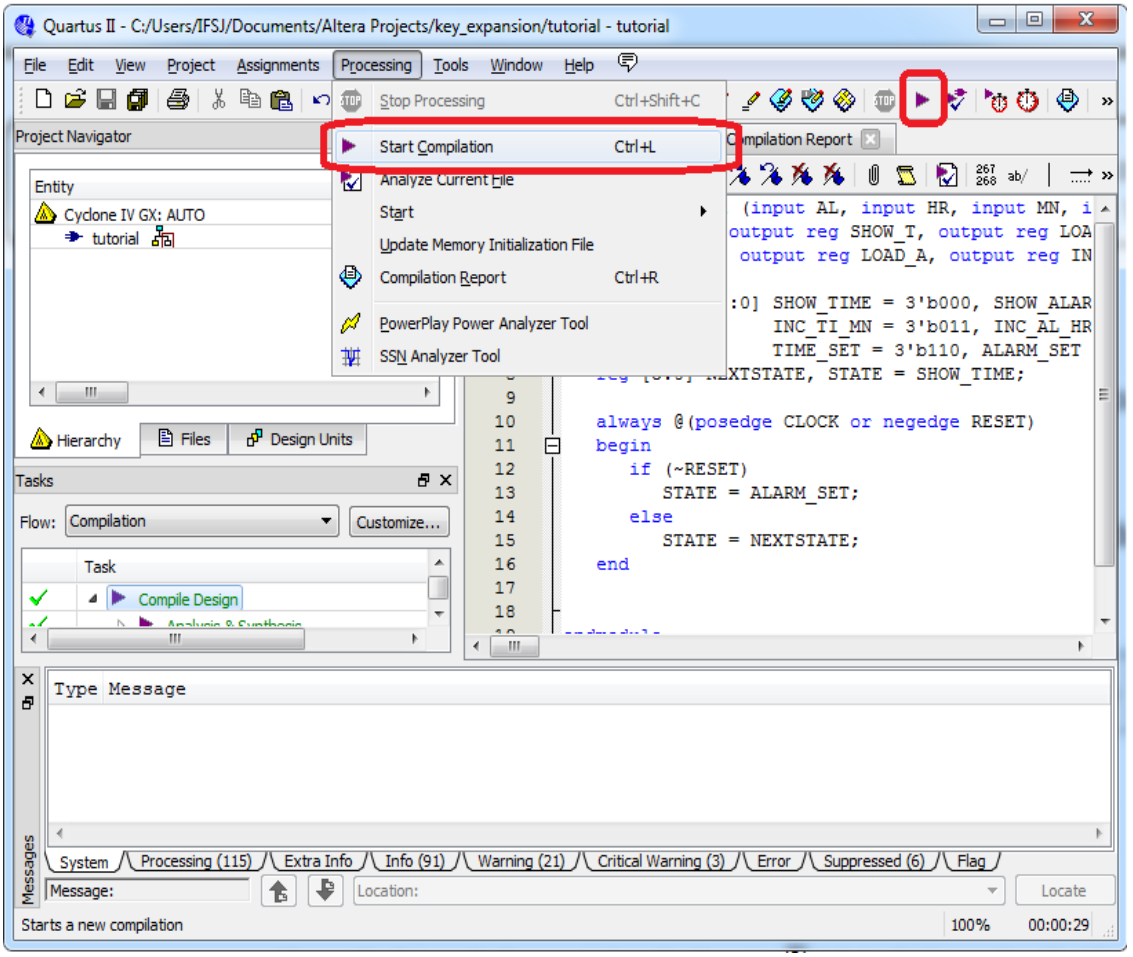


Figura 2.36. Tela indicando onde compilar a descrição SystemVerilog

Durante a compilação, percebem-se mudanças nas regiões **Task** e **Messages** do Quartus. Na primeira delas, é demonstrado como anda o processo de compilação, ou seja, em que fase está e qual a porcentagem do processo foi efetuada até o momento, enquanto que na aba **Messages** são exibidas as mensagens referentes à compilação, como a mudança de fase na compilação e principalmente os erros e warnings encontrados (quando existem), enquanto é exibido ainda o relatório da compilação (**Compilation Report**).

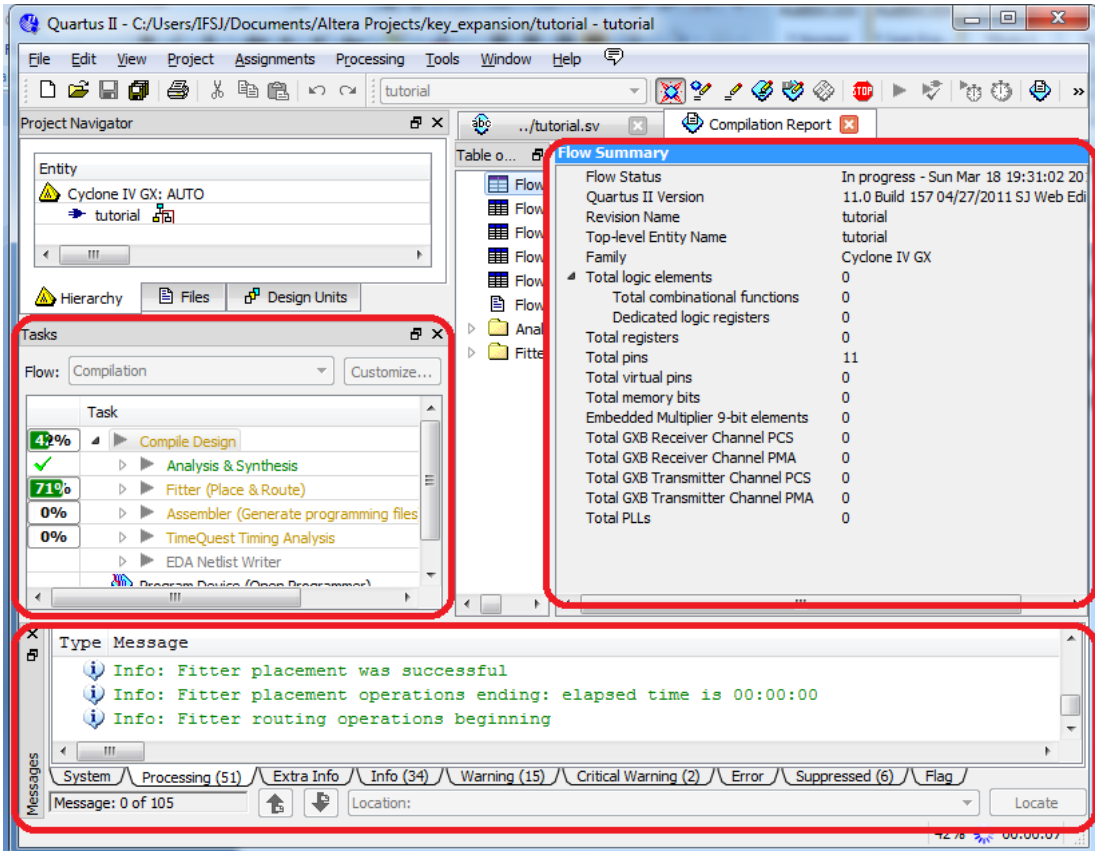


Figura 2.37. Tela de desenvolvimento do projeto

7. RTL Viewer

Para facilitar a análise e a depuração de um projeto, o Quartus II dispõe de alguns visualizadores gráficos de NETLIST sintetizados, que são resultados da compilação: RTL Viewer, State Machine Viewer e Technology Map Viewer. O RTL Viewer mostra o resultado da síntese a nível de transferências (Register Transfer Level – RTL), que consiste basicamente em uma representação por registradores interligados por lógica combinacional. Para obtê-lo, deve-se selecionar a opção **Tools > Netlist Viewers > RTL Viewer**.

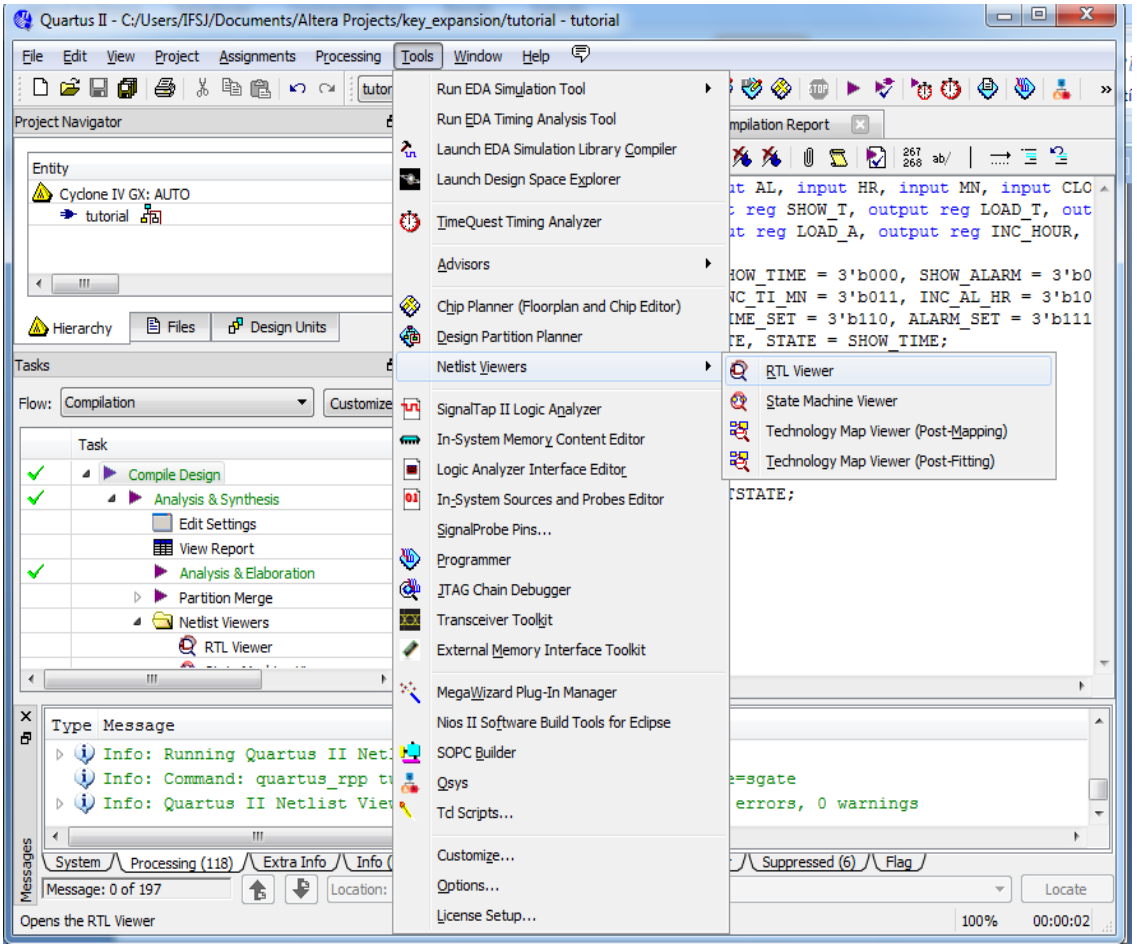


Figura 2.38. Tela mostrando onde encoontrar o RTL viewer para observar o esquemático

Quando existe mais de um arquivo no projeto, pode haver problemas com a **top-level Entity**, ou seja, com a entidade de nível mais alto. Tal entidade deve ser exatamente o arquivo que será usado como principal. Para torná-lo uma entidade deste tipo, deve-se abrir o arquivo, e **Project > Set as Top-Level Entity**.

2.5 Laboratórios

O objetivo desses laboratórios é o desenvolvimento de alguns hardwares simples, para introduzir SystemVerilog no desenvolvimento de circuitos integrados. Com tal finalidade, os laboratórios foram desenvolvidos para o desenvolvimento de circuitos sequenciais e combinacionais.

2.5.1 Laboratório 1

No laboratório 1 o aluno irá desenvolver um MUX 2X1 utilizando **case**, **if-else** e continous **assign.** Esse laboratório tem o objetivo de observar como uma mesma lógica pode ser sintetizada de formas diferentes.

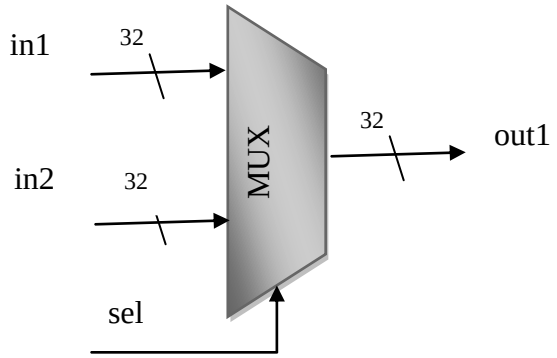


Figura 2.39. Mux 2X1

Após a descrição do código, o aluno deverá simular e observar o esquemático obtido através do RTL viewer.

2.5.2 Laboratório 2

Use SystemVerilog procedural para desenvolver um simples registrador. Nesse exercício você terá a oportunidade de explorar tipos de dados como **always_ff e iff.** Após o desenvolvimento, simule e observe o esquemático no RTL viewer. Como sinais de entrada teremos o **clock**, o **reset_n** (que é ativo na borda de descida) e um dado de um bit denominado **in1**. A saída do registrador **out1** deverá ser registrada.

2.5.3 Laboratório 3

Desenvolva um módulo que receba um byte como entrada e tenha duas saídas de quatro bits, MSB e LSB. Em seguida descreva um módulo decodificador que receba duas entradas de quatro bits e me retorne duas saídas de 32 bits seguindo a tabela 1.5. Utilize para esse exercício as palavras reservadas **always_comb** e **unique case.**

Tabela 2.1. Tabela de decodificação dos símbolos recebidos

Data symbol decimal	**Data symbol (binary) (*b*0, *b*1, *b*2, *b*3)**	**Chip values (c0 c1 ... c30 c31)**
0	0 0 0 0	1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0
1	1 0 0 0	1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0
2	0 1 0 0	0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0
3	1 1 0 0	0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1
4	0 0 1 0	0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1
5	1 0 1 0	0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0
6	0 1 1 0	1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1
7	1 1 1 0	1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1
8	0 0 0 1	1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1
9	1 0 0 1	1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1
10	0 1 0 1	0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1
Data symbol decimal	**Data symbol (binary) (*b*0, *b*1, *b*2, *b*3)**	**Chip values (c0 c1 ... c30 c31)**

11	1 1 0 1	0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0
12	1 1 0 0	0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0
13	1 1 0 1	0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1
14	0 1 1 1	1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0
15	1 1 1 1	1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0

2.5.4 Laboratório 4

Desenvolva um módulo que servirá de memória assíncrona e faz a alocação dos dados de entrada no endereço informado. O módulo tem como entradas os seguintes sinais: **w_clk**, **r_clk, w_data, w_ena, r_addr, r_ena, w_addr.** Apenas terá como saída o sinal **r_data.**

O objetivo de cada sinal é:

- w_clk – sinal referente ao clock para entrada de dados;
- r_clk – sinal referente ao clock para leitura de dados;
- w_data – barramento de 32 bits referente ao dado;
- w_ena – sinal que habilita o processo de escrita no módulo;
- (Opcional) r_ena – sinal que habilita o processo de leitura no módulo;
- r_addr – barramento que indica o endereço onde o dado será lido;
- w_addr – barramento que indica o endereço onde o dado será escrito;
- r_data – barramento de 32 bits referente ao dado de leitura do módulo;

2.6. Referências

Asic-World tutorial http://www.asic-world.com/systemverilog/tutorial.html

IEEE STANDARDS. *IEEE Standard for SystemVerilog* - **Unified Hardware Design, Specification, and Verification Language.** Institute of Electrical and Electronics Engineers, Inc., USA, 2005.

QuartusII tutorial Altera http://www.altera.com/literature/manual/mnl_qts_quick_start.pdf

Capítulo 3

Laboratório de Programação Paralela com MPI

Oberdan Rocha Pinheiro, André Luiz Lima da Costa e Josemar Rodrigues de Souza

Abstract

Despite advances in the field of computer science and particularly computing architectures, there are still many real problems that are difficult to solve in current computing platforms because of its high cost. The use of parallel computers in areas that require a high processing power is a reality, but the high cost of programming quality solutions further hinders the entry of emerging centers of scientific production in areas most advanced research. The objective of this lab is to teach programmers who are not familiar with the basics of MPI to develop and execute parallel programs in accordance with the rules of the MPI standard.

Resumo

Apesar dos avanços da área de ciência da computação e em particular das arquiteturas computacionais, ainda existem diversos problemas reais que são de difícil solução em plataformas computacionais correntes em virtude de seu alto custo. O uso de computadores paralelos em áreas que necessitam de um grande poder de processamento é uma realidade, mas o alto custo de programar soluções de qualidade ainda dificulta a entrada de centros emergentes de produção científica em áreas de pesquisa mais avançada. O objetivo do laboratório é ensinar aos programadores que não estão familiarizados com MPI (Message Passing Interface) as noções básicas para desenvolver e executar programas paralelos de acordo com as normas do padrão MPI.

3.1. Introdução

A partir o surgimento do primeiro computador digital eletrônico, em 1946, a computação passou por um processo evolutivo, em nível de hardware e software, com o objetivo de proporcionar maior desempenho e ampliar o leque de aplicações que podem ser computacionalmente resolvidas de maneira eficiente. Na primeira geração de computadores estabeleceram-se os conceitos básicos de organização dos computadores eletrônicos, e na década de 50 apareceu o modelo computacional que se tornaria a base

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

de todo o desenvolvimento subseqüente, chamado “modelo de von Neumann”.

O modelo de von Neumann é formado basicamente pelo conjunto: processador, memória e dispositivos de E/S. O processador executa instruções seqüencialmente de acordo com a ordem ditada por uma unidade de controle. Durante as décadas seguintes, houve um aperfeiçoamento muito grande dos componentes do modelo von Neumann, e este ciclo evolutivo permanece até os dias atuais.

Novas tecnologias de organização computacional começaram a ser desenvolvidas, em paralelo à evolução do modelo de von Neumann, buscando maior eficiência e facilidade no processo computacional. Dentre essas novas tecnologias, podemos citar a computação paralela, as redes de computadores e os mecanismos pipeline. As redes de computadores se tornaram mais rápidas e mais confiáveis, o que possibilitou a interligação dos computadores pessoais e estações de trabalho de maneira eficiente formando os sistemas distribuídos. Sistemas distribuídos são construídos com o intuito de oferecer uma imagem de sistema único, de maneira que o usuário não perceba que está trabalhando em vários computadores ao mesmo tempo, apesar de seus componentes estarem distribuídos.

Sistemas distribuídos têm sido utilizados para a execução de programas paralelos, em substituição às arquiteturas paralelas, em virtude de seu menor custo e maior flexibilidade. Nesse sentido, apesar de terem surgido por motivações diferentes, à computação paralela e a computação distribuída têm demonstrado certa convergência, visto que ambas possuem características e problemas semelhantes como: balanceamento de carga, modularidade e tolerância a falhas.

Pode-se observar que a computação paralela e a computação seqüencial de von Neumann, apesar de desenvolverem-se de maneira relativamente independente, atualmente têm apresentado uma relação de interseção significativa.

As sessões do Laboratório de Programação Paralela com MPI apresentam uma visão geral da computação paralela através da apresentação de vários conceitos e as noções básicas para desenvolver e executar programas paralelos de acordo com as normas do padrão MPI.

3.2. Estrutura Detalhada do Curso

3.2.1. Objetivos do Curso

O objetivo deste curso é ensinar aos programadores que não estão familiarizados com MPI as noções básicas para desenvolver e executar programas paralelos de acordo com as normas do padrão MPI. Os temas que serão apresentados terão como enfoque as rotinas básicas para programadores iniciantes em MPI.

3.2.2. Tipo de Curso

O Laboratório de Programação Paralela com MPI tem um foco prático, com o assunto dividido em um determinado número de sessões de aula, acompanhadas de um texto base, onde se resumem as noções tratadas no curso e se incluem uma serie de exercícios práticos a serem realizados pelos alunos.

O curso esta dividido em três sessões:

- Sessão 1 - Conceitos Fundamentais.
- Sessão 2 - Arquiteturas Paralelas.

- Sessão 3 - Metodologias de Paralelização de Algoritmos
- Sessão 4 - Introdução a Programação Paralela com MPI.
- Sessão 5 - Programas Exemplos

3.2.3. Estrutura prevista detalhada do texto

Nesta seção será descrita a estrutura do texto correspondente ao material didático do Laboratório de Programação Paralela com MPI.

- **Conceitos Fundamentais:** Essa sessão apresenta os conceitos fundamentais sobre paralelismo e concorrência, tipos e estilos de programação, nível de paralelismo, *speedup*, eficiência, escalabilidade e paradigmas de programação.
- **Arquiteturas Paralelas:** Essa sessão aborda os conceitos sobre a utilização de arquiteturas paralelas, pontos positivos e negativos, classificação de Flynn e classificação de Duncan.
- **Metodologias de Paralelização de Algoritmos:** Essa sessão apresenta uma visão geral sobre alguns dos principais conceitos associados aos clusters de computadores e as aplicações paralelas e distribuídas, abrangendo metodologias de paralelização.
- **Introdução a Programação Paralela com MPI:** Nessa seção serão discutidos alguns pontos importantes do modelo de programação baseado em passagem de mensagens. As características dos clusters e o padrão MPI e seus modos de comunicação. Dentre as primitivas disponibilizadas pelo padrão, descrevemos a sintaxe e a semântica das principais operações ponto-a-ponto e coletivas.
- **Programas Exemplos:** Nessa seção serão apresentados exemplos que tratam da sintaxe e programação das rotinas básicas do padrão MPI: Inicialização de processos MPI, identificando um processo no MPI, contando processos no MPI, enviando mensagens no MPI, recebendo mensagens no MPI e encerramento de um processo no MPI.

3.3. Conceitos Fundamentais

3.3.1. Paralelismo e Concorrência

Concorrência existe quando, em um determinado instante, dois ou mais processos começaram a sua execução, mas não terminaram. Por essa definição, concorrência pode ocorrer tanto em sistemas com um único processador, quanto em sistemas com múltiplos processadores.

Afirmar que processos estão sendo executados em paralelo implica na existência de mais de um processador, ou seja, paralelismo (ou paralelismo físico) ocorre quando há mais de um processo sendo executado no mesmo intervalo de tempo. Esse tipo de concorrência é demonstrado na Figura 3.1 que mostra a execução de três processos (e1, e2 e e3) em função do tempo.

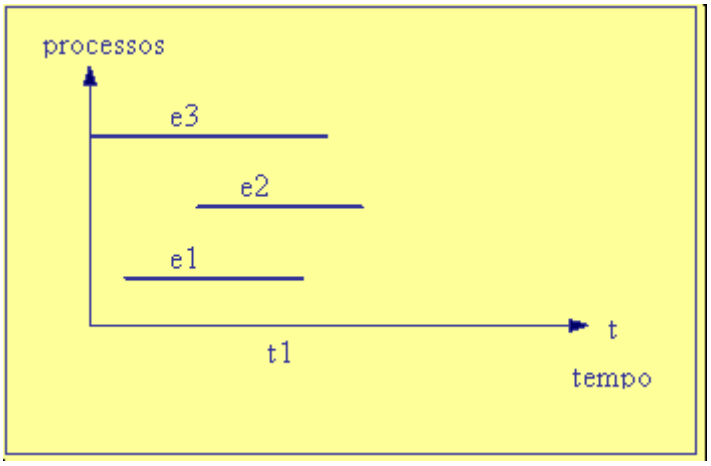


Figura 3.1. Paralelismo Físico

Quando vários processos são executados em um único processador, sendo que somente um deles é executado em cada vez, tem-se um pseudoparalelismo (paralelismo lógico). O usuário tem a falsa impressão de que suas tarefas estão sendo executadas em paralelo, mas, na realidade, o processador está sendo compartilhado entre os processos. Isso significa que, em um determinado instante, somente um processo está sendo executado, enquanto os outros que já foram iniciados aguardam a liberação do processador para continuarem sua execução (Figura 3.2).

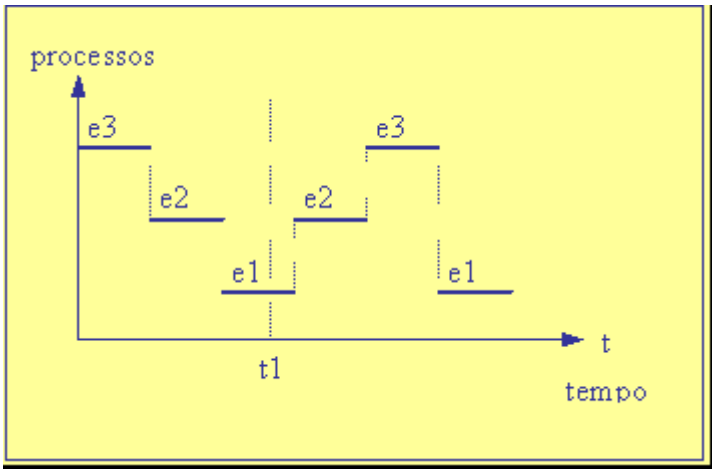


Figura 3.2. Paralelismo Lógico

Os tipos de estilos de programação dentro da computação são:

- **Programação seqüencial** - Caracteriza-se pela execução de várias tarefas uma após a outra;
- **Programação concorrente** - Caracteriza-se pela iniciação de várias tarefas, sem que as anteriores tenham necessariamente terminado (sistemas multi ou uniprocessadores);
- **Programação paralela** - Caracteriza-se pela iniciação e execução das tarefas em paralelo (sistemas multiprocessadores).

3.3.2. Nível de Paralelismo

O nível de paralelismo ou granulação está relacionado com tamanho das unidades de trabalho submetidas aos processadores. Esta definição é importante na computação

paralela, visto que está intimamente ligada ao tipo de plataforma (o porte e a quantidade de processadores) à qual se aplica o paralelismo. Algumas definições de granulação podem ser encontradas na literatura. Mas, de maneira simples, a granulação pode ser dividida em três níveis: grossa, fina e média.

Algumas definições sobre nível de paralelismo ou granulação podem ser encontradas na literatura. Mas, de maneira simples, a granulação pode ser dividida em três níveis:

- Granulação grossa relaciona o paralelismo em nível de processos e programas, e geralmente se aplica a plataformas com poucos processadores grandes e complexos.
- Granulação fina, por outro lado, relaciona paralelismo em nível de instruções ou operações e implica em um grande número de processadores pequenos e simples.
- A granulação média situa-se em um patamar entre as duas anteriores, implicando em procedimentos sendo executados em paralelo.

3.3.3. Desempenho em Programas Paralelos

O tempo de execução de um algoritmo paralelo depende de duas variáveis: o tamanho do problema e o número de processos utilizados para resolver o problema [PACHECO 1996]. Quando o desempenho de um programa paralelo é discutido, geralmente, usa-se a comparação com o tempo obtido com a versão serial. As métricas comumente utilizadas na computação paralela são s*peedup* e eficiência.

3.3.3.1. Speedup

O *Speedup* (1) é definido como a razão entre o tempo de execução da solução serial do algoritmo e o tempo de execução da solução paralela do algoritmo [PACHECO 1996].

$$speedup = \frac{T1}{Tp} \quad (1)$$

Onde, T1 representa o tempo de execução serial da solução e Tp representa o tempo execução da solução paralela em p processos. Pacheco (1996) diz que alguns autores consideram o T1 como o algoritmo serial mais rápido para resolver o problema executado em um dos processos do cluster. No entanto, Pacheco (1996) considera que não é fácil definir qual é o mais rápido algoritmo serial para resolver determinado problema. Por conta disso, uma solução utilizada é executar o algoritmo serial no processo mais rápido e utilizar esse tempo como base.

Idealmente, o ganho de *speedup* deveria tender a p, que seria o seu valor ideal 1. Porém, alguns fatores influenciam nessa relação impedindo que o *speedup* ideal seja alcançado. Pelo menos três desses fatores podem ser citados: a sobrecarga de comunicação entre os processadores, partes do código executável estritamente seqüencial e o nível de paralelismo utilizado (em virtude do uso de granulação inadequada à arquitetura).

3.3.3.2. Eficiência

A eficiência (2) é definida como a razão entre o *speedup* (1) e a quantidade de p processos utilizados para a execução do algoritmo paralelo.

$$\text{eficiência} = \frac{speedup}{p} \tag{2}$$

No caso ideal (*speedup* = p), a eficiência seria máxima e teria valor 1 (100%).

3.3.4. Escalabilidade

A escalabilidade é outra métrica utilizada para avaliar a performance dos algoritmos paralelos. Pacheco (1996) diz que um programa paralelo é escalável no caso de, ao se aumentar o número de processos p, encontra-se uma razão de incremento para o tamanho do problema n, de forma que a eficiência continue constante.

A escalabildade de uma aplicação paralela reflete a habilidade da utilização efetiva dos recursos computacionais incrementados [GRAMA et al. 2003]. Ou seja, essa métrica verifica se os recursos disponíveis adicionados de fatos estão sendo utilizados. Logo, pode-se afirmar que se tem um algoritmo escalável, se for possível obter uma eficiência, no mínimo, constante, com o aumento simultâneo de processadores e do tamanho do problema.

3.3.5. Paradigmas de Programação

Buyya (1999), afirma que as aplicações paralelas podem ser classificadas em alguns poucos paradigmas definidos. Cada paradigma é uma classe de algoritmo que possui a mesma estrutura de controle. A escolha do paradigma de programação paralela é determinada pela disponibilidade dos recursos computacionais e pelo tipo de paralelismo do problema. Os recursos computacionais podem ser definidos pelo nível da granularidade que pode ser efetivamente suportada pelo sistema. O tipo do paralelismo reflete a estrutura da aplicação ou de dados e ambos os tipos podem existir em diferentes partes da mesma aplicação.

Buyya (1999) define os paradigmas de programação paralela da seguinte forma:

- *Master*/*Worker* - Nesse paradigma, o *Master* é responsável por decompor o problema em diversas tarefas menores e distribuir essas tarefas entre os *Workers*, além de obter e ordenar os resultados parciais a fim de produzir o resultado final da computação. O papel dos *Workers* é obter a tarefa enviada na mensagem, processar a tarefa e mandar os resultados para o *Master*. Normalmente, a comunicação ocorre somente entre o *Master* e os *Workers*;
- *Single Program Multiple Data* (SPMD) - Cada processo executa basicamente o mesmo código, porém em partes diferentes de dados. Existe a comunicação entre os processos vizinhos;
- *Data Pipelining* - As tarefas do algoritmo que são capazes de operações concorrentes são identificadas e cada processador executa uma pequena parte do algoritmo total;
- *Divide and Conquer* - Um problema é dividido em dois ou mais subproblemas. Cada um desses subproblemas é resolvido independentemente e seus resultados são combinados para se obter o resultado final.

3.4. Arquiteturas Paralelas

Uma arquitetura paralela fornece uma estrutura explícita e de alto nível para o desenvolvimento de soluções utilizando o processamento paralelo, através da existência de múltiplos processadores, simples ou complexos, que cooperam para resolver problemas através de execução concorrente.

3.4.1. Programação Paralela

A computação paralela requer a utilização de outro paradigma de programação. No paradigma serial, o fluxo de execução segue uma ordem serial, com eventuais mudanças nesses fluxos quando ocorre uma chamada à função ou alguma instrução de repetição.

No paradigma paralelo, o algoritmo é executado paralelamente por diversos processos. Segundo Dongarra et al. (2003), a principal mudança nesse paradigma é que o programa deve ser decomposto em sub-componentes que podem ser executados paralelamente.

3.4.2. Classificação de Flynn

Segundo Flynn (1966), o processo computacional deve ser visto como uma relação entre fluxos de instruções e fluxos de dados. Um fluxo de instruções equivale a uma seqüência de instruções executadas (em um processador) sobre um fluxo de dados aos quais estas instruções estão relacionadas. A taxonomia proposta por Flynn (1966) é composta por quatro categorias:

- SISD - Single Instruction, Single Data. São arquiteturas que possuem uma única unidade de controle e que produzem um único fluxo de instruções. Essa classe é representada pela máquina seqüencial de Von Neumann, conforme Figura 3.3.

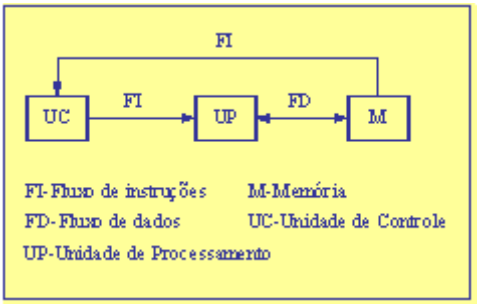


Figura 3.3. Modelo SISD

- SIMD - Single Instruction Stream/Multiple Data Stream (Fluxo único de instruções/Fluxo múltiplo de dados). Envolve múltiplos processadores (escravos) sob o controle de uma única unidade de controle (mestre), executando simultaneamente a mesma instrução em diversos conjuntos de dados conforme Figura 3.4. Arquiteturas SIMD são utilizadas, por exemplo, para manipulação de matrizes e processamento de imagens.

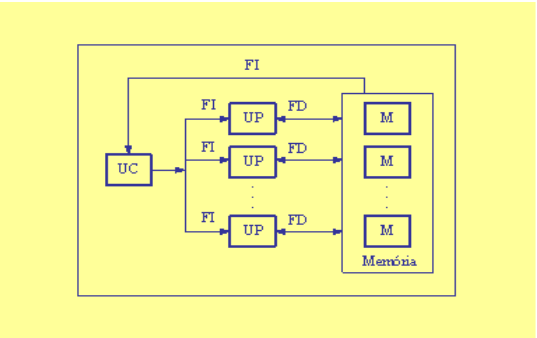


Figura 3.4. Modelo SIMD

- MISD - Multiple Instruction Stream/Single Data Stream (Fluxo múltiplo de instruções/Fluxo único de dados). Envolve múltiplos processadores executando diferentes instruções em um único conjunto de dados, conforme. Essa é uma arquitetura teórica.
- MIMD - Multiple Instruction Stream/Multiple Data Stream (Fluxo múltiplo de instruções/Fluxo múltiplo de dados). Envolve múltiplos processadores executando diferentes instruções em diferentes conjuntos de dados, de maneira independente conforme Figura 3.5. Esta classe engloba a maioria dos computadores paralelos.

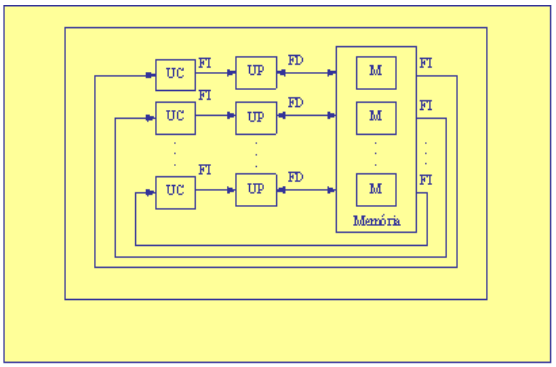


Figura 3.5. Modelo MIMD

A classificação de Flynn apresenta alguns problemas. Ela não é abrangente o suficiente para incluir alguns computadores modernos (por exemplo, processadores vetoriais e máquinas de fluxo de dados). A fim de acrescentar novas arquiteturas paralelas que surgiram, sem descartar a classificação de Flynn, Duncan (1990) propôs uma classificação mais completa que permite apresentar uma visão geral dos estilos de organização para computadores paralelos da atualidade.

3.4.3. Classificação de Duncan

A classificação de Duncan surgiu para representar arquiteturas mais recentes que não podem ser associadas com precisão as categorias escritas na taxonomia proposta por Flynn. Essa classificação divide as arquiteturas em dois grupos principais: arquiteturas síncronas e assíncronas, conforme apresentado na Figura 3.6.

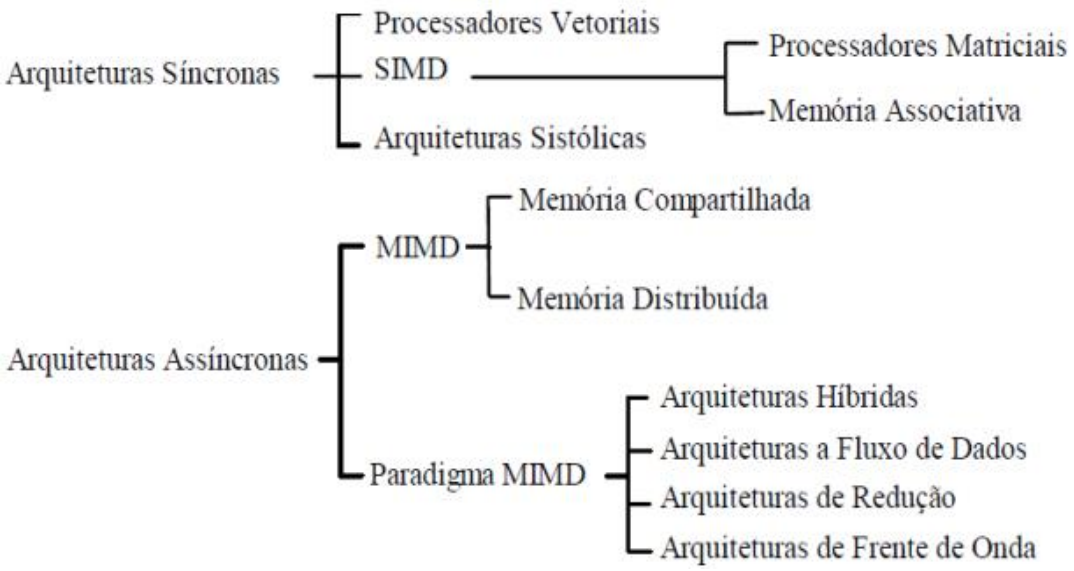


Figura 3.6. Classificação de Duncan.

Arquiteturas paralelas síncronas coordenam suas operações concorrentes sincronamente em todos os processadores, através de relógios globais, unidades de controle únicas ou controladores de unidades vetoriais. Tais arquiteturas apresentam pouca flexibilidade para a expressão de algoritmos paralelos.

- Processadores Vetoriais - Caracterizam-se pela existência de um hardware específico para a execução de operações em vetores. Essas operações são implementadas através de instruções vetoriais. Os processadores vetoriais *pipeline* são caracterizados por possuírem múltiplas unidades funcionais organizadas de maneira *pipeline*, onde são implementadas as instruções vetoriais. Essas arquiteturas foram criadas com o intuito de prover mecanismos eficientes para o suporte de cálculos pesados em matrizes ou vetores.
- Arquiteturas SIMD: Essas arquiteturas apresentam múltiplos processadores, sob a supervisão de uma unidade central de controle, que executam a mesma instrução sincronamente em conjuntos de dados distintos. Podem ser organizadas de duas maneiras:
    - Processadores Matriciais: projetados especialmente para fornecer estruturas para computação sobre matrizes de dados, são empregados para fins específicos. Fornecem acesso à memória via endereço, o que os diferencia do modelo de memória associativa. Exemplos: Illiac IV e IBM GF11.
    - Memória Associativa: relaciona arquiteturas SIMD cujo acesso à memória é feito de acordo com o seu conteúdo, em contraste ao método de acesso usual, via endereço. O esquema associativo visa permitir o acesso paralelo à memória, de acordo com um padrão de dados. Exemplos: Goodyear Aerospace STARAN e Parallel Element Processing Esemble (PEPE).
- Arquiteturas Sistólicas: propostas no início da década de 80, por H.T. Kung na Universidade de Carnegie Mellon, estas arquiteturas têm como principal objetivo fornecer uma estrutura eficiente para a solução de

problemas que necessitem de computação intensiva junto à grande quantidade de operações de E/S. Essas arquiteturas se caracterizam pela presença de vários processadores, organizados de maneira *pipeline*, que formam uma cadeia na qual apenas os processadores localizados nos limites desta estrutura possuem comunicação com a memória. Desta maneira, o conjunto de dados percorre toda a cadeia de processadores, de forma rítmica e sincronizada por um relógio global, não havendo armazenamento temporário em memória na comunicação entre os processadores.

Arquiteturas Assíncronas caracterizam-se pelo controle descentralizado de hardware, de maneira que os processadores são independentes entre si. Esta classe é formada basicamente pelas arquiteturas MIMD, convencionais ou não.

- Arquiteturas MIMD: são arquiteturas compostas por vários processadores independentes, onde se executam diferentes fluxos de instruções em dados locais a esses processadores.
    - Arquiteturas de memória compartilhada (ou multiprocessadores) caracterizam-se pela existência de uma memória global e única, que é utilizada por todos os processadores (sistema fortemente acoplado), de maneira que através do compartilhamento de posições desta memória ocorre a comunicação entre processos. Para evitar que a memória se torne um gargalo, arquiteturas de memória compartilhada devem implementar mecanismos de *cache*, além de garantir a coerência dos dados (através de mecanismos de hardware e software).
    - Em arquiteturas de memória distribuída, cada processador possui sua própria memória local, caracterizando assim um sistema fracamente acoplado. Em virtude de não haver compartilhamento da memória, os processos comunicam-se via troca de mensagens. Nesse tipo de comunicação ocorre a transferência explícita de dados entre os processadores. Este tipo de organização é também conhecido como multicomputador.
- Paradigma MIMD: essa classe engloba as arquiteturas assíncronas que, apesar de apresentarem a característica de multiplicidade de fluxo de dados e instruções das arquiteturas MIMD, são organizadas segundo características tão particulares ao seu projeto quanto suas características MIMD, o que impede que essas arquiteturas sejam classificadas como puramente MIMD. Por isso, tais arquiteturas se denominam paradigmas arquiteturais MIMD.
    - Arquiteturas MIMD/SIMD (híbridas), também conhecidas por MSIMD, caracterizam-se por apresentarem controle SIMD para determinadas partes de uma arquitetura MIMD. A flexibilidade que este modelo pode apresentar é atrativa (por exemplo, alguns nós podem ser processadores vetoriais). As aplicações nas quais elas podem ser utilizadas são: processamento de imagens e sistemas especialistas.
    - Arquiteturas de Redução, também conhecidas como arquiteturas dirigidas a demanda, baseiam-se no conceito de redução, que

implica que partes do código fonte original sejam reduzidas aos seus resultados em tempo de execução. As instruções são ativadas para serem executadas quando os seus resultados são necessários como operandos por outra instrução já ativada para execução.

A classificação de Duncan engloba a maioria dos tipos de arquiteturas existentes, tendo atingido os itens abrangência e extensibilidade, nos quais a classificação de Flynn falha. Além disso, uma característica importante da classificação proposta por Duncan é a presença dos termos MIMD e SIMD, propostos por Flynn e largamente aceitos.

3.5. Metodologias de Paralelização de Algoritmos

Desenvolver um algoritmo seqüencial para solucionar um problema computacional é o modo convencional de ensino de programação e comumente adotado na prática. A forma de abstrair o processo de um programa tendo um início, meio e fim, no qual uma etapa é realizada somente após o termino da anterior, ocorre entre o início do seu processo até o seu término [Evans e Goscinski, 1995].

Foster (1995) define que um algoritmo paralelo deve considerar quatro aspectos explanados a seguir:

- Concorrência: capacidade de duas ou mais instruções serem executadas simultaneamente;
- Escalabilidade: capacidade de oferecer suporte ao aumento do número de processadores;
- Modularidade: habilidade de decompor componentes complexos em entidades mais simples permitindo até mesmo a reusabilidade pregada pela engenharia de software, tanto no desenvolvimento de aplicações seqüenciais quanto nas paralelas;
- Localidade: é a razão entre a comunicação da memória local e remota. Esta característica deve ser fortemente considerada em ambientes multicomputadores em que o barramento de comunicação entre processos é uma rede de interconexão cuja latência é inúmeras vezes maior que o barramento memória-processador.

Geralmente o algoritmo paralelo é desenvolvido a partir de uma implementação seqüencial. Durante a análise do código seqüencial, o programador deve encontrar os blocos que não possuem dependência entre sí. Se um resultado não tem dependência de outros anteriores, então a paralelização é possível, otimizando o desempenho da aplicação.

Foster (1995) descreve uma metodologia para auxiliar no processo de desenvolvimento de uma aplicação paralela denominada PCAM . Esta metodologia está organizada em quatro fases distintas: particionamento, comunicação, aglomeração e mapeamento. Os quatro estágios estão ilustrados na Figura 3.7 e são descritos a seguir:

- Particionamento: nesta primeira etapa, é realizada a decomposição tanto das operações quanto os dados do problema em pequenas tarefas. Não considera as questões práticas como o número de processadores ou a arquitetura em que será executada, mas sim em detectar os potenciais pontos de paralelização no código;

- Comunicação: Nesta etapa são determinadas as comunicações necessárias para a coordenação da execução das tarefas, definindo os algoritmos e as estruturas de dados necessárias;
- Aglomeração: Os artefatos gerados nos estágios anteriores são avaliados sob os requisitos de desempenho e custos de implementação. Caso necessário, as tarefas são combinadas em tarefas maiores para melhorar o desempenho e reduzir custos de desenvolvimento;
- Mapeamento: Nesta última etapa, cada tarefa é distribuída a um processador de maneira que esta aproveite os recursos de processamento e com baixo custo de comunicação. A distribuição das tarefas pode ser estática ou determinada por um algoritmo de balanceamento de carga durante a execução da aplicação.

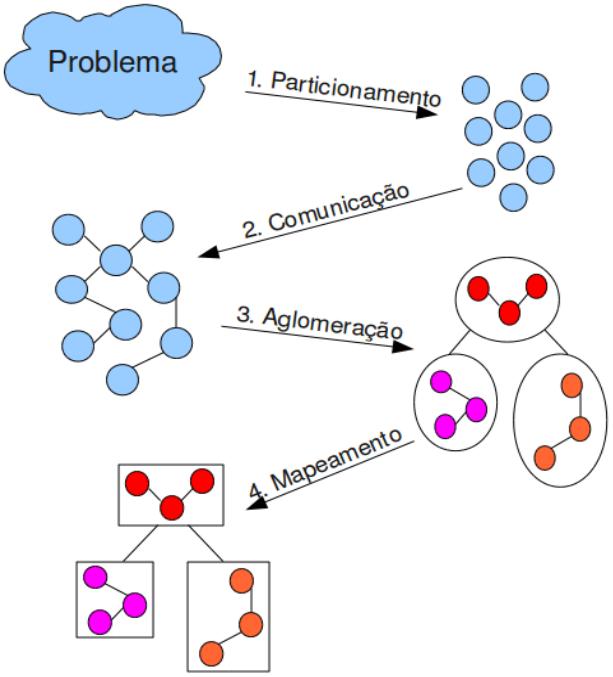


Figura 3.7. Etapas da metodologia PCAM

3.5.1. Granularidade de Aplicações

A granularidade pode ser considerada fina quando o paralelismo está presente em nível de instruções ou grossa quando em nível de aplicações [Hayes et al., 1992]. Porém, a granularidade de um algoritmo pode ocorrer também em níveis intermediários, podendo envolver outros fluxos de execução como threads, funções ou estruturas lógicas do algoritmo, de modo que podemos então definir a granularidade como uma medida relativa. Quanto mais fina, as instruções executadas pelo processador são mais rápidas, mas o tráfego no barramento de comunicação será maior, resultando na saturação do sistema de comunicação e na queda do desempenho da aplicação. Deste modo, quanto maior a granularidade apresentada, melhor será o desempenho obtido, visto que o *overhead* da comunicação será reduzido. As operações em estruturas matriciais são fáceis de paralelizar por não haver dependência sobre as linhas e colunas, o qual permite cada processador efetuar as operações de forma isolada para que, posteriormente, possam ser agrupadas pelo computador mestre.

Os termos mais usados para esta classificação são o paralelismo *coarse-grain* e *fine-grain* [Tanenbaum e Woodhull 2006], [Silberchatz et al. 2008]. Esses termos estão relacionados ao nível e freqüência da comunicação e sincronização que ocorre dentro de um programa paralelo. O termo *fine-grain* (granularidade fina) representa um conjunto de dados em um nível mais baixo. O termo baixo nível neste contexto quer dizer que o nível de paralelismo é baseado em instruções mais próximas das instruções de máquina [Tanenbaum e Woodhull 2006]. As instruções do código fonte podem ser executadas em paralelo sendo que cada instrução no código possui poucas instruções para serem enviadas. O paralelismo *coarse-grain* (granularidade grossa) é baseado conceitualmente em uma abstração mais alta do que o paralelismo de granularidade fina. Esse paralelismo está acima da estrutura lógica do programa. Normalmente, é implementado em procedimentos, funções e alguns outros tipos de agrupamento de códigos.

A granularidade fina e grossa pode ser comparada à execução de máquinas forte e fracamente acopladas, respectivamente. Tanenbaum e Woodhull (2006) observam que a granularidade fina normalmente é executada em arquiteturas fortemente-acopladas enquanto o paralelismo de granularidade grossa é usualmente encontrado em máquinas fracamente-acopladas. Neste sentido, o algoritmo da aplicação deste laboratório se encaixa na classificação de granularidade grossa e utiliza a linguagem C e a biblioteca Open MPI para o desenvolvimento e execução das versões paralelas.

3.6. Introdução a Programação Paralela com MPI

A necessidade de aplicações capazes de oferecer melhor desempenho tem impulsionado o desenvolvimento de tecnologias na área de computação paralela. Uma forma de melhorar este desempenho consiste na elaboração de técnicas e ferramentas adequadas às arquiteturas das máquinas paralelas, sejam elas de memória compartilhada ou de memória distribuída. Através da correta exploração das características das máquinas paralelas e do paralelismo implícito nas aplicações, os pesquisadores procuram a obtenção de menores tempos de execução. O aperfeiçoamento das bibliotecas de passagem de mensagem, por exemplo, pode ajudar projetistas de sistemas a explorarem melhor este modelo de programação e obterem programas mais eficientes.

O modelo de programação baseado em passagem de mensagens requer o uso de uma biblioteca de funções oferecendo suporte à comunicação entre diferentes processos ou nós de processamento. Atualmente, o MPI (*Message Passing Interface*) [MPI Forum] tem se consolidado como um padrão para o desenvolvimento de aplicações baseadas em trocas de mensagens. O padrão MPI propõe uma biblioteca de funções específicas, permitindo a realização de diversas operações de comunicação, e possui diversas implementações disponíveis.

3.6.1. Programação Paralela Através de Passagem de Mensagens

O projeto e a implementação de uma aplicação paralela costumam ser realizados segundo um modelo de programação, que por sua vez deve refletir as características da máquina a ser utilizada para executar a aplicação. Segundo Culler (1999) descreve os principais modelos de programação paralela da seguinte forma:

- Endereços compartilhados: as comunicações são executadas através de espaços de endereçamento compartilhados, cujo uso é análogo à utilização de um quadro de avisos onde as pessoas podem escrever e ler

informações. Este modelo de programação é muitas vezes também denominado de memória compartilhada;

- Passagem de mensagens: na programação através de passagem de mensagens, um conjunto de eventos possibilita as trocas de informações entre emissores e receptores específicos, não existindo uma memória compartilhada. Tais eventos constituem a base para harmonizar atividades individuais;
- Processamento paralelo de dados: neste modelo, mais conhecido como processamento vetorial, vários agentes executam uma ação simultânea sobre elementos distintos de um conjunto de dados. Após a execução da ação, os dados podem ser reorganizados através de acessos a endereços compartilhados ou trocas de mensagens.

As operações de comunicação mais comuns em sistemas de passagem de mensagens são o *send* e o *receive*, incluindo suas diversas variantes possíveis. Como sugerem os nomes, o aparecimento de um *send* em um programa indica a realização de uma troca de mensagem entre dois processos, sob a condição de existir uma respectiva operação *receive* para a concretização da comunicação. A combinação destas operações determina um evento de sincronização entre os processos participantes e permite a cópia de dados do espaço de endereçamento do processo emissor para o espaço de endereçamento do receptor.

Existem diversas variantes possíveis para as operações *send/receive*. Dependendo de quando o *send* é finalizado em relação à execução do *receive*, ou do momento em que o *buffer* do emissor é disponibilizado para reuso ou, ainda, de quando a requisição é aceita, implementações distintas destas operações podem determinar diferentes eventos de sincronização [CULLER 1999]. Assim, diferentes semânticas e requisitos de implementação são necessários de acordo com as possíveis variantes do *send/receive*.

3.6.1.1. Características dos Clusters

Nos últimos anos os sistemas clusters têm sido vistos como soluções viáveis para a construção de aplicações paralelas e distribuídas, tanto na comunidade acadêmica como no meio empresarial. O uso de clusters na solução de problemas que exigem características como grande poder computacional e alta disponibilidade tem merecido destaque e ganho a atenção de diversos grupos de pesquisa. Isso se deve às vantagens apresentadas por esta arquitetura, dentre as quais podemos destacar:

- *Hardware* de baixo custo e fácil aquisição: clusters podem ser construídos a partir de estações de trabalho ou de computadores pessoais ligados através de uma rede de alta velocidade;
- Componentes de *software* padronizados: as bibliotecas de suporte à programação baseada em trocas de mensagens, como o MPI e o PVM (Parallel Virtual Machine), e o uso de sistemas operacionais não proprietários, como o Linux, facilitam e barateiam o desenvolvimento de aplicações para serem executadas em clusters;
- Escalabilidade: dependendo da aplicação, podemos facilmente inserir ou retirar máquinas do sistema computacional e obtermos significativos ganhos de desempenho. A inserção e remoção de nós de processamento muitas vezes não acarreta em grandes alterações nos programas ou na

maneira como estes são desenvolvidos. Além disso, com um custo relativamente pequeno, podemos obter significativas melhorias no desempenho de programas paralelos adicionando novos nós de processamento ao cluster.

Segundo Moura e Silva (1999), um cluster é caracterizado como um sistema de processamento paralelo ou distribuído, composto por um conjunto de computadores distintos trabalhando como um recurso computacional único e integrado. Dependendo da configuração das máquinas constituintes do sistema, este sistema de processamento pode ser dito homogêneo ou heterogêneo, sendo a primeira classificação atribuída àqueles sistemas compostos por máquinas idênticas, enquanto um sistema heterogêneo é formado por máquinas de configurações distintas. Desta forma, cada nó de processamento pode ser uma máquina mono ou multiprocessada com sua própria memória local, dispositivos de entrada e saída e um sistema operacional.

A Figura 3.8 [MOURA e SILVA 1999] ilustra as principais características da arquitetura de um sistema cluster. Como podemos observar, um conjunto de máquinas conectadas através de uma rede local de alta velocidade e um switch são utilizadas para executar aplicações seqüenciais e paralelas. Cada uma destas máquinas, por exemplo, um PC ou uma estação de trabalho, possui sua própria interface de rede, responsável por receber e transmitir pacotes de dados através da rede de interconexão. Garantir a fidelidade dos dados transmitidos é uma das funções do software de comunicação, que está diretamente ligado aos processos de empacotamento e desempacotamento das mensagens transmitidas.

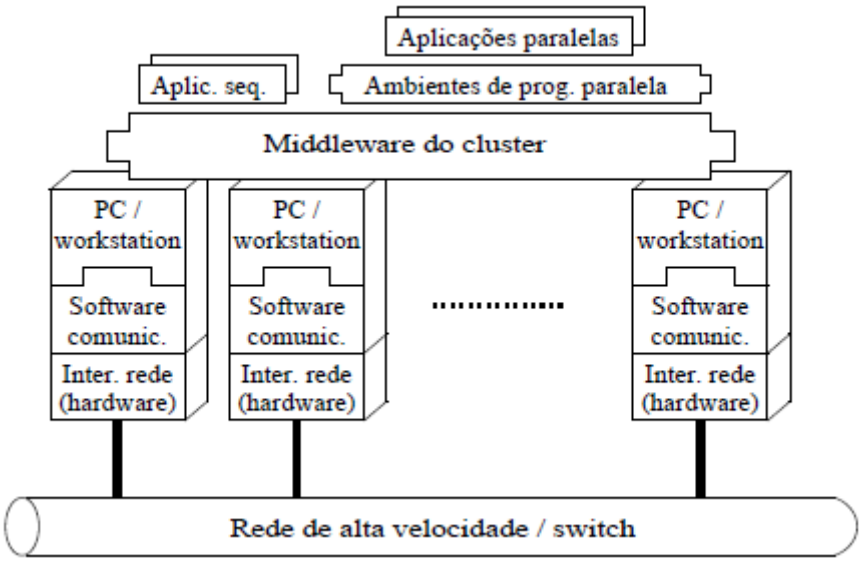


Figura 3.8. Arquitetura de um cluster.

A criação da imagem de um sistema de processamento integrado e único é uma característica comumente observada em clusters. Tal imagem é gerada com o suporte de um *middleware*, que não apenas possibilita o processamento paralelo e distribuído, mas também permite a execução de aplicações seqüenciais. Desta forma, as máquinas podem trabalhar como um sistema computacional único ou como computadores individuais.

Para possibilitar a execução de programas paralelos, é necessário ainda um ambiente de programação adequado. Estes ambientes podem envolver, por exemplo, bibliotecas de passagens de mensagens.

3.6.1.2. Implementação do Modelo de Passagem de Mensagens

Na implementação das trocas de mensagens, os detalhes de movimentação de dados costumam ser escondidos do programador em uma biblioteca de passagem de mensagens. Uma camada de software, inserida entre as camadas das primitivas de comunicação e o hardware do sistema, possibilita ao programador construir aplicações paralelas sem precisar preocupar-se com pequenos detalhes envolvidos nas transferências de dados.

Em [CULLER 1999], são discutidos alguns aspectos de baixo nível referentes à implementação das operações de trocas de mensagens, que apresentamos a seguir. Embora estas operações pudessem ser implementadas diretamente por hardware, suas características (como, por exemplo, bufferização) são mais bem tratadas por implementação de software. Assim, em todas as máquinas paralelas, o modelo de programação através de passagem de mensagens é realizado com uma camada de software construída sobre uma abstração de comunicação mais simples. Primitivas mais básicas de transferência de dados podem ser utilizadas para implementar tal abstração, em uma solução suportada diretamente por hardware. Outra forma de implementação desta abstração seria considerarmos a adoção de um espaço de endereçamento virtual compartilhado, permitindo que as comunicações sejam realizadas através de operações de escrita e leitura em buffers compartilhados e envolvam eventos de sincronização apropriados [CULLER 1999].

3.6.1.3. Sincronização e Endereçamento

O modelo de programação através de passagem de mensagens tem sido bastante utilizado no desenvolvimento de aplicações paralelas, principalmente em máquinas de memória distribuída. Geralmente, todos os processos executam cópias idênticas de um único programa [CULLER 1999] e as trocas de mensagens constituem um poderoso mecanismo de sincronização entre processos cooperantes.

Um exemplo de sincronização implementada com a utilização de mecanismos de trocas de mensagens trata-se das operações *send/receive*. Cada par *send/receive* pode estabelecer um evento de sincronização em um determinado ponto do programa envolvendo os processos emissor e receptor. Operações de comunicação coletivas também podem permitir a sincronização de processos, dependendo da implementação realizada.

Conforme é discutido em [CULLER 1999], em uma máquina de passagem de mensagens, um processador pode referenciar apenas endereços em sua memória local e cada um dos demais processadores. Assim, é permitido a um processo usuário acessar seus endereços privados e transferir dados usando primitivas de comunicação. Cada processo possui seu espaço de endereçamento privado, onde são realizadas as operações locais segundo a ordem de execução do programa.

3.6.1.4. Modos de Comunicação

Conforme discutido anteriormente, as operações mais utilizadas em sistemas baseados em trocas de mensagens são o *send* e o *receive*. Devido ao fato de envolverem apenas dois processos, um enviando e outro recebendo dados, estas operações são chamadas ponto-a-ponto.

Em diversas situações surge a necessidade de um processo enviar ou receber dados de vários processos, como por exemplo, no caso em que um mestre distribui

informações ou recebe resultados de seus escravos. Operações de comunicação coletiva podem ser utilizadas com este intuito, permitindo transferências de dados nos sentidos um para vários, vários para um e vários para vários. A seguir listamos alguns exemplos de comunicações coletivas:

- Um-para-vários: *broadcast* e *scatter*;
- Vários-para-um: *gatter* e *reduce*;
- Vários-para-vários: *all-to-all*.

Uma comunicação *send/receive* pode ainda ser bloqueante ou não-bloqueante. Considerando uma operação de envio, por exemplo, costumamos caracterizá-la como bloqueante se o processo emissor for impedido de prosseguir sua execução até que o buffer de dados utilizado para transmitir a mensagem possa ser reutilizado. O uso de operações bloqueantes pode levar a ocorrência de deadlocks, sendo responsabilidade do programador evitar tais situações, contudo as não-bloqueantes são menos seguras pois permitem o reuso de buffers que ainda estão sendo utilizados em uma comunicação. Uma diferenciação mais clara destes modos de comunicação é apresentada a seguir [MPI FORUM 1995]:

- Bloqueante: o retorno da primitiva indica que o usuário pode reutilizar com segurança os recursos especificados na chamada (como buffers);
- Não bloqueante: a primitiva pode retornar antes que a operação de comunicação complete e antes que o usuário possa reutilizar os recursos especificados na chamada.

3.6.1.5. Bibliotecas de Suporte às Trocas de Mensagens

No modelo de programação paralela através de passagem de mensagens, costuma-se utilizar bibliotecas apropriadas como suporte à comunicação entre os processos. Bibliotecas ou interfaces de passagem de mensagens, como muitas vezes também são denominadas, possibilitam a construção de programas paralelos eficientes para sistemas de memória distribuída [MOURA e SILVA 1999].

Tais bibliotecas oferecem diversas rotinas para suportar a inicialização e finalização do ambiente e o envio e recebimento de pacotes de dados. Além das tradicionais comunicações *send* e *receive*, algumas variações destas operações costumam ser implementadas, disponibilizando formas de comunicação ponto-a-ponto com diferentes eventos de sincronização e, em alguns casos, introduzindo o uso de buffers para a execução das trocas de mensagens. As operações de comunicação coletiva e as barreiras constituem outros importantes mecanismos de trocas de informações oferecidos. Em síntese, podemos dizer que geralmente as bibliotecas de passagem de mensagem implementam operações ponto-a-ponto e coletivas, bloqueantes e não-bloqueantes.

As interfaces mais utilizadas na programação através de trocas de mensagens são o PVM (Parallel Virtual Machine) e o padrão MPI (Message Passing Interface) [MPI FORUM 1995], definido pelo Fórum MPI. Ambas procuram oferecer ao programador o suporte necessário para o desenvolvimento de aplicações paralelas eficientes baseadas em trocas de mensagens e possuem implementações para as linguagens C, C++ e Fortran, além de Java cujo suporte têm sido desenvolvido [PRAMANICK 1999]. Contudo, tarefas de paralelização como comunicação e sincronização entre processos, particionamento e distribuição de dados e mapeamento dos processos entre os

processadores disponíveis, ainda ficam encarregadas ao programador [MOURA e SILVA 1999].

A maior desvantagem do PVM em relação ao MPI trata-se do desempenho, fato que por vezes estimula projetistas e pesquisadores a escolherem a segunda interface. Para conseguir maior flexibilidade, o PVM acaba sacrificando seu desempenho [PRAMANICK 1999]. O melhor desempenho do MPI tem impulsionado o crescimento de sua popularidade e atualmente este é o padrão adotado pela maioria dos centros de pesquisa e fabricantes.

3.6.1.6. Desempenho das Comunicações

Um dos fatores que influenciam fortemente o desempenho das aplicações paralelas baseadas em trocas de mensagens refere-se ao tempo despendido com as comunicações. O desempenho das comunicações, por sua vez, depende de diversos elementos constituintes do sistema computacional (como as características da rede de interconexão e das máquinas envolvidas).

A Figura 3.2 [HENNESSY 1996] ilustra os parâmetros de desempenho de uma rede de interconexão. O overhead de envio e o overhead de recepção de uma mensagem indicam as frações de tempo em que os processadores envolvidos na comunicação, emissor e receptor respectivamente, ficam dedicados às tarefas de disponibilizar e retirar a mensagem da rede de interconexão. Durante este tempo os processadores não realizam outros processamentos.

O tempo necessário para que o primeiro bit alcance a interface de rede da máquina de destino é indicado pelo componente na figura denominado como time of flight. A soma deste tempo com o tempo de transmissão, durante o qual o restante da mensagem passa através da rede, constitui a latência de transporte. Na Figura 3.9 podemos observar melhor este comportamento, além de visualizar a latência total da comunicação.

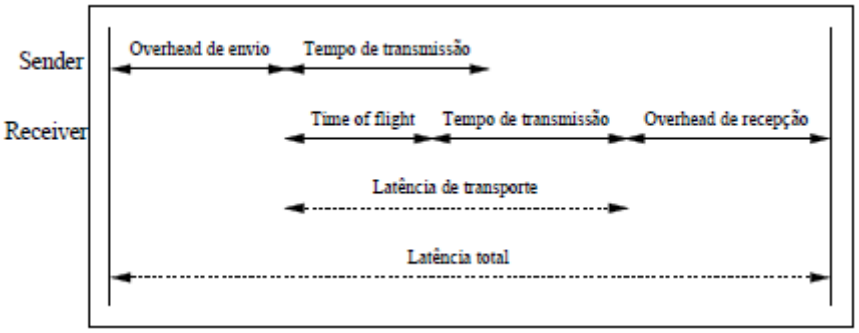


Figura 3.9. Parâmetros de desempenho de uma rede de interconexão.

Tratando-se do desempenho de redes de interconexão, um dos termos mais utilizados é a largura de banda (bandwidth). Geralmente medida em megabits por segundos (Mbits/s), a largura de banda refere-se à taxa máxima segundo a qual a rede pode propagar informação e está diretamente relacionada com o tempo de transmissão de uma mensagem. Assumindo que não existam outras mensagens concorrendo pelo uso da rede, o tempo de transmissão equivale ao tamanho da mensagem dividido pela largura de banda.

Um erro por vezes observado em projetos de avaliação de desempenho trata-se da adoção da largura de banda como a única medida de desempenho da rede. Porém, conforme é comentado em [HENNESSY 1996], para muitas aplicações e redes, a maior

influência na latência de comunicação de uma mensagem é exercido, sobretudo pelos overheads de envio e recepção.

3.6.2. Características Básicas do Padrão MPI

Até alguns anos atrás o desenvolvimento de aplicações paralelas e distribuídas baseadas em trocas de mensagens não se mostrava uma solução atraente aos fabricantes de software. Um dos motivos disto relaciona-se à diversidade de bibliotecas disponíveis que implementavam o modelo de programação baseado em passagem de mensagens, sem contudo existir um mesmo padrão aceito e utilizado pelos projetistas.

Neste contexto, o PVM e o padrão MPI propiciaram novos impulsos à disseminação do uso de passagem de mensagens na construção de aplicações para máquinas de memória distribuída. O padrão MPI, criado mais recentemente, possibilita a diferentes fabricantes de software utilizar um mesmo conjunto de primitivas de comunicação que, embora possam apresentar diferentes implementações, são definidas segundo uma mesma sintaxe e semântica. As implementações do padrão obviamente possuem características internas distintas, porém disponibilizam a mesma interface de passagem de mensagens.

O MPICH [GROPP,W. et AL 1996] e o LAM [LAM/MPI 2012] são as mais conhecidas implementações do padrão MPI. No entanto, variações significativas de desempenho podem ou não ser verificadas entre implementações diferentes, dependendo de fatores como as métricas adotadas ou as características do ambiente de teste.

Tabela 3.1. Tipos de dados MPI e seus correspondentes na linguagem C.

MPI	C
MPI_CHAR	signed char
MPI_SHORT	signed short
MPI_INT	signed int
MPI_LONG	signed long
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long
MPI_FLOAT	Float
MPI_DOUBLE	Double
MPI_BYTE	-
MPI_PACKED	-

3.6.2.1. Tipos de Dados MPI

A comunicação de uma mensagem MPI envolve a transmissão de uma determinada quantidade de dados de um tipo pré-definido. Desta forma, em uma operação de comunicação o usuário necessita indicar o número de elementos transmitidos e não a respectiva quantidade de bytes da mensagem, o que torna esta tarefa independente das características da máquina utilizada, pois não é requerido o conhecimento do tamanho em bytes dos elementos envolvidos na comunicação. A Tabela 3.2 apresenta os principais tipos de dados MPI e seus correspondentes na linguagem C [MPI FORUM 1995].

Os dois últimos tipos apresentados na Tabela 3.1 não possuem correspondentes na linguagem C. Enquanto cada elemento de dado MPI_BYTE é composto exatamente por um 1 byte, o MPI_PACKED na realidade constitui um tipo de dados derivado e permite ao usuário empacotar elementos homogêneos não contínuos em um único buffer

que será enviado. Para o empacotamento e posterior desempacotamento dos dados, realizado após a recepção da mensagem, o padrão MPI apresenta as respectivas operações MPI_pack() e MPI_unpack(). Maiores informações sobre outros tipos de dados derivados podem ser encontrados na definição do padrão MPI [MPI FORUM 1995].

3.6.2.2. Comunicadores MPI

Segundo a definição do padrão MPI [MPI FORUM 1995], os seguintes requisitos são esperados em uma biblioteca paralela robusta:

- Criação de um espaço de comunicação seguro, garantindo a execução de operações de comunicação sem que ocorram conflitos com comunicações não relacionadas à biblioteca;
- Permitir o uso de escopo de grupo para facilitar a realização de comunicações coletivas;
- Identificação dos processos usuários através de nomes abstratos;
- Permitir ao usuário ou construtor da biblioteca estender notações do modelo de passagem de mensagens.

Com vistas à satisfação destes requisitos, o padrão MPI inclui entre suas definições os conceitos de communicators, contextos de comunicação e grupos de processos. Em um programa MPI, uma operação de comunicação é executada dentro de um determinado contexto, que é especificado por um communicator e indica os possíveis processos receptores da mensagem.

Um grupo de processos constitui uma coleção ordenada de processos, onde cada um destes possui uma identificação única (um número inteiro, denominado rank, iniciando em 0). Um processo pode pertencer a mais de um grupo e, neste caso, possuir diferentes ranks.

O conceito de contexto permite particionar o espaço de comunicação, de forma que a operação de envio de uma mensagem possa ser direcionada a um grupo específico de processos. Mensagens enviadas dentro de um contexto não podem ser recebidas em outro. Além disso, convém ressaltar que contextos não constituem objetos MPI, mas precisam ser indicados em todas as primitivas de comunicação através de um parâmetro denominado communicator.

Um communicator trata-se de um objeto MPI que provê o escopo apropriado para a execução de uma operação de comunicação MPI. Tal operação pode ser executada dentro de um único grupo de processos, utilizando um intra-communicator, ou pode constituir uma comunicação ponto-a-ponto envolvendo dois grupos distintos (indicada pelo uso de um inter-communicator). Podemos citar como exemplos de intra-communicators pré-definidos pelo padrão o MPI_COMM_WORLD e o MPI_COMM_SELF, sendo a abrangência do primeiro todos os processos envolvidos na execução do programa MPI e a do segundo apenas o próprio processo.

3.6.2.3. Primitivas de Comunicação MPI

3.6.2.3.1. Comunicação Ponto-a-Ponto

O padrão MPI disponibiliza um conjunto de primitivas para realizar operações de comunicação ponto-a-ponto bloqueantes e não-bloqueantes (Tabela 3.2). Tais primitivas diferenciam-se principalmente por implementarem diferentes mecanismos de sincronismo entre os processos envolvidos (o receptor e o emissor), ou pelo uso de buffers no processo de transmissão da mensagem.

Como podemos observar na Tabela 3.2, os parâmetros utilizados nas primitivas bloqueantes (MPI_Send, MPI_Bsend, MPI_Rsend e MPI_Ssend) são exatamente os mesmos. Em uma operação de comunicação ponto-a-ponto MPI, o processo emissor envia um número de elementos (count) de um determinado tipo (dtype) para o processo receptor (dest). Os dados a serem enviados devem estar no buffer de envio buf e a variável dest identifica o receptor dentro do grupo de processos indicado pelo communicator comm. Existe ainda outro importante parâmetro, chamado tag, que pode ser usado como uma identificação da mensagem transmitida.

Tabela 3.2. Operações de comunicação ponto-a-ponto MPI.

Operação	Primitiva MPI
Send padrão	MPI_Send(&buf, count, dtype, dest, tag, comm);
Send buffered	MPI_Bsend(&buf, count, dtype, dest, tag, comm);
Send ready	MPI_Rsend(&buf, count, dtype, dest, tag, comm);
Send síncrono	MPI_Ssend(&buf, count, dtype, dest, tag, comm);
Send não bloqueante	MPI_Isend(&buf, count, dtype, dest, tag, comm, &req);
Receive bloqueante	MPI_Recv(&buf, count, dtype, source, tag, comm, &status);
Receive não bloqueante	MPI_Irecv(&buf, count, dtype, source, tag, comm, &req);

A seguir podemos visualizar melhor estes parâmetros em uma chamada da primitiva MPI_Send:

MPI_Send(void * buf, int count, MPI_Datatype dtype, int dest, int tag, MPI_Comm comm)

O send padrão trata-se do modo de comunicação mais utilizado para o envio de uma mensagem entre dois processos MPI. A operação é dita bloqueante pois, ao ser executada, ela apenas poderá retornar após a mensagem ter sido armazenada com segurança, sendo então permitido ao processo emissor reutilizar o buffer de envio [MPI Forum]. Ainda segundo as definições do Fórum MPI, a mensagem pode ser transmitida diretamente para o buffer do receptor ou simplesmente ser copiada em um buffer de sistema, sendo realmente transferida ao seu destino em um segundo momento. É de responsabilidade do MPI decidir quando os dados envolvidos em uma comunicação são ou não armazenados localmente; por isso, podemos dizer que a primitiva de comunicação pode ou não bloquear dependendo de sua implementação.

Conforme comentado em [AL-TAWIL 2001], o comportamento do send padrão não é definido precisamente pelo MPI. Assim, se pretendemos compreender o que realmente ocorre durante a transmissão de uma mensagem neste modo de comunicação, precisamos analizar as particulariedades da implementação MPI utilizada. Após estudarmos o código da implementação LAM, verificamos que a primitiva MPI_Send pode ou não bloquear dependendo, dentre outros fatores, do tamanho da mensagem transmitida e do número de mensagens pendentes a serem recebidas pelo processo de destino.

A principal diferença do buffered send em relação aos demais modos de send está no uso de um buffer de dados para onde a mensagem é copiada antes de ser transmitida. O usuário deve criar este buffer explicitamente, através de funções específicas disponibilizadas pelo MPI, antes de chamar a primitiva MPI_Bsend. É de responsabilidade do usuário definir um buffer suficientemente grande para alojar a

mensagem transmitida e cuidar para que este não seja reutilizado indevidamente, evitando a ocorrência de erros ou a sobreposição de dados.

O tempo despendido pelo processo emissor em uma comunicação no modo buffered tende a ser menor em relação aos demais sends bloqueantes. Isso acontece, pois, ao contrário das operações MPI_Send, MPI_Ssend e MPI_Rsend, a ocorrência de uma primitiva MPI_Bsend em um programa não exige a existência de uma operação de recepção (MPI_Recv) para a sua correta finalização. A mensagem é simplesmente copiada para o buffer, de onde será transmitida ao respectivo receptor, e o processo emissor pode continuar sua execução normalmente. Por outro lado, a necessidade da alocação deste buffer gera um overhead que pode ser significativo dependendo do tamanho da mensagem. Assim, caso o objetivo do usuário ao utilizar o MPI_Bsend seja obter melhor desempenho, talvez uma alternativa melhor possa ser oferecida pelo send não-bloqueante.

A operação MPI_Ssend pode ser inicializada com ou sem a ocorrência do respectivo MPI_Recv, mas somente poderá ser finalizada quando o receptor começar a receber a mensagem enviada [TENNESSEE 2012]. Desta forma, o MPI garante que ao final da execução da operação de envio o processo receptor atingiu um ponto de sua execução e o buffer do emissor pode ser reutilizado [MPI FORUM 1995].

No modo ready do send bloqueante a operação de comunicação somente terá início quando houver um receptor esperando a mensagem. Quando ocorre uma chamada da primitiva MPI_Rsend, o MPI verifica a existência da respectiva operação MPI_Recv e, caso a resposta seja negativa, a comunicação fica aguardando. Caso contrário, a transmissão da mensagem é estabelecida e uma vez encerrada a primitiva será finalizada. Convém notarmos a diferença entre os eventos de sincronismo estabelecidos pelo modo síncrono e ready. Enquanto no primeiro temos a transferência dos dados e em seguida a sincronização dos processos, no ready os processos devem estar sincronizados antes do início da transmissão da mensagem em si.

Os quatro modos de comunicação ponto-a-ponto bloqueante discutidos nesta seção requerem a mesma primitiva para a recepção dos dados, apresentada abaixo:

MPI_Recv(void * buf, int count, MPI_Datatype dtype, int source, int tag, MPI_Comm comm, MPI_Status * status)

Conforme é comentado na definição do padrão MPI [MPI FORUM 1995], a primitiva MPI_Recv não exige a finalização do respectivo send para completar sua execução. Porém, é claro que ela somente terá início após a inicialização da operação send.

3.6.2.3.2. Comunicação Coletiva

As primitivas descritas a seguir oferecem suporte às comunicações coletivas em um programa MPI. São classificadas como operações coletivas aquelas envolvendo um grupo de processos [MPI FORUM 1995]. Na Tabela 3.3 podemos conferir algumas destas operações e as primitivas MPI correspondentes.

Tabela 3.3. Principais operações de comunicação coletiva MPI.

Operação	Primitiva MPI
Broadcast	MPI_Bcast(&buf, count, dtype, root, comm);
Scatter	MPI_Scatter(&sendbuf, sendcount, sendtype, &recvbuf, recvcount, recvtype, root, comm);

Gather	MPI_Gather(&sendbuf, sendcount, sendtype, &recvbuf, recvcount, recvtype, root, comm);
Reduce	MPI_Reduce(&sendbuf, &recvbuf, count, dtype, op, root, comm);
Barreira	MPI_Barrier(comm);
All-to-all	MPI_Alltoall(&sendbuf, sendcount, sendtype, &recvbuf, recvcount, recvtype, comm);

Enquanto nas operações ponto-a-ponto o emissor e o receptor precisam realizar chamadas de primitivas distintas (por exemplo, MPI_Ssend e MPI_Recv, respectivamente), nas operações coletivas os processos envolvidos utilizam a mesma primitiva. A distinção entre qual é o processo emissor (ou emissores) e quais são os receptores (ou receptor) é dada pelo parâmetro root, segundo a semântica da operação. O grupo de processos para o qual a operação coletiva é direcionada, por sua vez, é indicado através do parâmetro comm, especificando o communicator utilizado.

Conforme é discudo na definição do padrão [MPI FORUM 1995], a finalização de uma primitiva coletiva indica que o buffer utilizado na comunicação pode ser acessado com segurança, embora não necessariamente todos os processos envolvidos tenham completado a operação (ou até mesmo a iniciado). Esta informação não é válida para a operação MPI_Barrier, onde ocorre explicitamente a sincronização dos processos. Porém, para os demais casos, sugere que podemos ter ou não ter um evento de sincronismo associado à ocorrência da comunicação coletiva. As respectivas semânticas do broadcast, do scatter, do gather e do reduce podem ainda ser observadas na Figura 3.10.

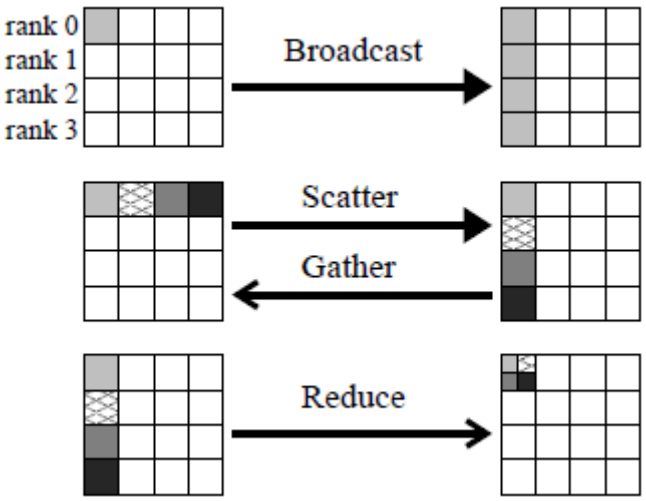


Figura 3.10. Operações de comunicação coletiva.

A operação broadcast, cuja primitiva MPI é apresentada abaixo, caracteriza-se como uma comunicação coletiva do tipo um-para-vários, onde o processo root envia uma mesma mensagem a todos os processos do grupo, inclusive ele mesmo. Assim como ocorre com as demais operações coletivas, é comum o uso de operações ponto-a-ponto em sua implementação e a forma de distribuição dos dados pode variar dependendo do algoritmo utilizado.

MPI_Bcast(void * buf, int count, MPI_Datatype dtype, int root, MPI_Comm comm)

Nas operações scatter e gather as mensagens transmitidas são diferentes, ao contrário do que ocorre com o broadcast. As comunicações são classificadas, respectivamente, como um-para-vários e vários-para-um. Quando o MPI_Scatter é executado, o root divide a quantidade de dados sendcount pelo número de processos pertencentes ao grupo e então envia mensagens diferentes a todos os participantes do

grupo, inclusive a ele mesmo. Para os demais processos, os parâmetros do emissor (sendbuf, sendcount e sendtype) não são significantes.

A semântica da operação gather é o oposto do scatter. Assim, cada processo envia sendcount elementos do tipo sendtype ao root, que recebe os dados e os armazena segundo a ordem dos ranks. Convém ressaltar que tanto no MPI_Scatter como no MPI_Gather os tipos denotados pelos parâmetros sendtype e recvtype devem ser compatíveis e o número de elementos enviados é sempre igual ao de recebidos. Ambas as primitivas requerem exatamente os mesmos parâmetros. Abaixo apresentamos suas respectivas definições:

MPI_Scatter(void * sendbuf, int sendcount, MPI_Datatype sendtype, void * recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

MPI_Gather(void * sendbuf, int sendcount, MPI_Datatype sendtype, void * recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

A primitiva MPI_Reduce estabelece uma comunicação coletiva do tipo vários-para-um, onde o processo root recebe dados enviados pelos demais integrantes do grupo e realiza uma operação de redução sobre estes dados. Todos os processos devem especificar a mesma operação de redução através do parâmetro op. Alguns exemplos são: calcular a soma ou o produto dos elementos, encontrar o mínimo ou o máximo e as operações lógicas AND e OR.

MPI_Reduce(void * sendbuf, void * recvbuf, int count, MPI_Datatype dtype, MPI_Op op, int root, MPI_Comm comm)

Uma barreira pode ser criada explicitamente em um programa MPI através da primitiva MPI_Barrier, permitindo a sincronização de todos os processos do grupo. Ao executar a primitiva, o processo fica bloqueado até que todos os processos do grupo especificado pelo communicator comm tenham realizado uma chamada a esta primitiva.

MPI_Barrier(MPI_Comm comm)

O all-to-all constitui uma operação de comunicação vários-para-vários e, em sua execução, cada processo envia partes distintas de um conjunto de dados para os demais integrantes do grupo, conforme Figura 3.11 [MPI FORUM 1995]. A quantidade de elementos que compõe a mensagem é dividida pelo número de participantes da comunicação (X) e, em seguida, cada uma destas partes é destinada ao buffer de um receptor.

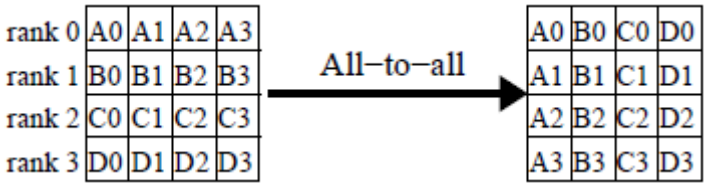


Figura 3.11. Operação coletiva all-to-all.

MPI_Alltoall(void * sendbuf, int sendcount, MPI_Datatype sendtype, void * recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm)

3.7. Programas Exemplos

3.7.1. MPI_INIT

Inicializa um processo MPI. Portanto, deve ser a primeira rotina a ser chamada por cada processo, pois estabelece o ambiente necessário para executar o MPI. Ela também sincroniza todos os processos na inicialização de uma aplicação MPI.

- Sintaxe:

- int MPI_Init (int *argc, char *argv[])

- Parâmetros:
  - argc - Apontador para a qtde. de parametros da linha de comando;
  - argv - Apontador para um vetor de strings.

- Erro:
  - MPI_ERR_OTHER - Ocorreu mais de uma chamada dessa rotina no mesmo código.

O código a seguir ilustra o funcionamento da função MPI_Init.

```
#include <stdio.h>
#include "mpi.h"
main(int argc, char *argv[]){
  int ret;
  ret = MPI_Init(&argc, &argv);
  if (ret < 0){
        printf ("Nao foi possivel inicializar o processo MPI!\n");
        return;
  }else{
        ...
  }
}
```

3.7.2. MPI_COMM_RANK

Identifica um processo MPI dentro de um determinado grupo. Retorna sempre um valor inteiro entre 0 e n-1, onde n é o número de processos.

- Sintaxe:
  - int MPI_Comm_rank (MPI_Comm comm, int *rank)

- Parâmetros:
  - comm - Comunicador do MPI;
  - rank - Variável inteira com o numero de identificação do processo.

- Erro:
  - MPI_ERR_COMM - Communicator inválido.

O código a seguir ilustra o funcionamento da função MPI_Comm_rank.

```
#include <stdio.h>
#include "mpi.h"
main(int argc, char *argv[]){
  int mpierr, rank;
  mpierr = MPI_Init(&argc, &argv);
  if (mpierr < 0){
        printf ("Nao foi possivel inicializar o processo MPI!\n");
        return;
  }else{
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

```
        printf ("Minha identificação no MPI e':%d\n",rank);
        ...
  }
}
```

3.7.3. MPI_COMM_SIZE

Retorna o número de processos dentro de um grupo.

- Sintaxe:
  - int MPI_Comm_size (MPI_Comm comm, int *size)
- Parâmetros:
  - comm - Comunicador do MPI;
  - size - Variável interna que retorna o numero de processos iniciados pelo MPI.
- Erros:
  - MPI_ERR_COMM - Communicator inválido.
  - MPI_ERR_ARG - Argumento inválido.
  - MPI_ERR_RANK - Identificação inválida do processo.

O código a seguir ilustra o funcionamento da função MPI_Comm_size.

```
#include <stdio.h>
#include "mpi.h"
main(int argc, char *argv[]){
  int mpierr, rank, size;
  mpierr = MPI_Init(&argc, &argv);
  if (mpierr < 0){
        printf ("Nao foi possivel inicializar o processo MPI!\n");
        return;
  }else{
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
        printf ("Minha identificação no MPI e':%d\n",rank);
        MPI_Comm_size(MPI_COMM_WORLD, &size);
        printf("O numero de processos e': %d\n",size);
        ...
  }
}
```

3.7.4. MPI_SEND

Rotina básica para envio de mensagens no MPI utiliza o modo de comunicação "blocking send" (envio bloqueante), o que traz maior segurança na transmissão da mensagem. Após o retorno, libera o "system buffer" e permite o acesso ao "application buffer".

- Sintaxe:
  - int MPI_Send (void *sndbuf, int count, MPI_Datatype dtype, int dest, int tag, MPI_Comm comm)

- Parâmetros:
  - sndbuf - Identificação do buffer (endereço inicial do "application buffer" - de onde os dados serão enviados);
  - count - Número de elementos a serem enviados;
  - dtype - Tipo de dado;
  - dest - Identificação do processo destino;
  - tag - Rótulo (label) da mensagem;
  - comm - MPI Communicator.
- Erros:
  - MPI_ERR_COMM - Communicator inválido;
  - MPI_ERR_COUNT - Argumento numérico inválido;
  - MPI_ERR_RANK - Identificação inválida do processo;
  - MPI_ERR_TYPE - Tipo de dado inválido;
  - MPI_ERR_TAG - Declaração inválida do label;
  - MPI_ERR_RANK - Identificação inválida do processo.

O código a seguir ilustra o funcionamento da função MPI_Send.

```
#include <stdio.h>
#include "mpi.h"
main(int argc, char *argv[]){
  int mpierr, rank, size, i;
  char message[20];
  mpierr = MPI_Init(&argc, &argv);
  if (mpierr < 0){
        printf ("Nao foi possivel inicializar o processo MPI!\n");
        return;
  }else{
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
        printf ("Minha identificação no MPI e':%d\n",rank);
        MPI_Comm_size(MPI_COMM_WORLD, &size);
        printf("O numero de processos e': %d\n",size);
        if (rank == 0){
          strcpy(message,"Ola', Mundo!\n");
          for (i=1; i<size; i++){
            MPI_Send(message, 13, MPI_CHAR, i, tag, MPI_COMM_WORLD);
          }
        }else{
                ...
        }
  }
}
```

3.7.5. MPI_RECV

É uma rotina básica para recepção de mensagens no MPI, que utiliza o modo de comunicação "*blocking receive*" (recepção bloqueante), de forma que o processo espera até que a mensagem tenha sido recebida. Após o retorno, libera o "*system buffer*", que pode ser então, novamente utilizado.

- Sintaxe:
  - int MPI_Recv (void *recvbuf, int count, MPI_Datatype dtype, int source, int tag, MPI_Comm comm, MPI_Status status)
- Parâmetros:
  - sndbuf - Identificação do buffer (endereço inicial do "application buffer" - de onde os dados estão sendo enviados);
  - count - Número de elementos a serem recebidos;
  - dtype - Tipo de dado;
  - source - Identificação do processo emissor;
  - tag - Rótulo (label) da mensagem;
  - comm - MPI Communicator
  - status - Vetor de informações envolvendo os parâmetros source e tag.
- Erros:
  - MPI_ERR_COMM - Communicator inválido.
  - MPI_ERR_COUNT - Argumento numérico inválido.
  - MPI_ERR_RANK - Identificação inválida do processo.
  - MPI_ERR_TYPE - Tipo de dado inválido.
  - MPI_ERR_TAG - Declaração inválida do label.
  - MPI_ERR_RANK - Identificação inválida do processo.

O código a seguir ilustra o funcionamento da função MPI_Recv.

```
#include <stdio.h>
#include "mpi.h"
main(int argc, char *argv[]){
  int mpierr, rank, size, tag, i;
  MPI_Status status;
  char message[20];
  mpierr = MPI_Init(&argc, &argv);
  if (mpierr < 0)        {
        printf ("Nao foi possivel inicializar o processo MPI!\n");
        return;
  }else{
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
        printf ("Minha identificação no MPI e':%d\n",rank);
        MPI_Comm_size(MPI_COMM_WORLD, &size);
        printf("O numero de processos e': %d\n",size);
        if (rank == 0){
          strcpy(message,"Ola', Mundo!\n");
```

```
        for (i=1; i<size; i++){
          MPI_Send(message, 13, MPI_CHAR, i, tag, MPI_COMM_WORLD);
        }
      }else{
        MPI_Recv(message, 20, MPI_CHAR, 0, tag, MPI_COMM_WORLD,
&status);
        printf("Mensagem do no' %d : %s\n", rank, message);
      ...
      }
  }
}
```

3.7.6. MPI_FINALIZE

Finaliza um processo MPI. Portanto deve ser a última rotina a ser chamada por cada processo. Sincroniza todos os processos na finalização de uma aplicação MPI.

- Sintaxe:
  - MPI_Finalize (void)
- Parâmetros:
  - Não se aplica.
- Erro:
  - Não se aplica.

O código a seguir ilustra o funcionamento da função MPI_Recv.

```
#include <stdio.h>
#include "mpi.h"
main(int argc, char *argv[]){
      int mpierr, rank, size, tag, i;
      MPI_Status status;
      char message[20];
      mpierr = MPI_Init(&argc, &argv);
      if (mpierr < 0){

      printf ("Nao foi possivel inicializar o processo MPI!\n");
                        return;
      }else{
            MPI_Comm_rank(MPI_COMM_WORLD, &rank);
            printf ("Minha identificação no MPI e':%d\n",rank);
        MPI_Comm_size(MPI_COMM_WORLD, &size);
        printf("O numero de processos e': %d\n",size);
        if (rank == 0){
              strcpy(message,"Ola', Mundo!\n");
          for (i=1; i<size; i++){
                MPI_Send(message, 13, MPI_CHAR, i, tag,
MPI_COMM_WORLD);
                                                        }
        }else{
```

```
                MPI_Recv(message, 20, MPI_CHAR, 0, tag,
MPI_COMM_WORLD, &status);
                printf("Mensagem do no' %d : %s\n", rank, message);
        MPI_Finalize();
        }
}
```

3.8. Referências

AL-TAWIL, K.; MORITZ, C. A. Performance Modeling and Evaluation of MPI. Journal of Parallel and Distribuited Computing, 61, p.202-223, 2001.

BUYYA, R. High Performance Cluster Computing: Architectures and Systems. [S.l.]: Prentice Hall PTR, 1999.

CULLER, D. E.; SINGH, J. P.; GUPTA, A. Parallel Computer Architecture: A Hardware/Software Approach. San Francisco, California: Morgan Kaufmann Publishers, 1999. 1025p.

DONGARRA, J. et al. Sourcebook of parallel computing. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003. ISBN 1-55860-871-0.

DUNCAN, R., A Survey of Parallel Computer Architectures, IEEE Computer, p. 5-16, Fevereiro, 1990.

EVANS, D.; Goscinski, A. A Survey of Basic Issues of Parallel Execution on a Distributed System. Relatório Técnico, Citeseer, 1995.

FOSTER, I. Designing and building parallel programs: concepts and tools for parallel software engineering. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1995.

FLYNN, M. Very-high speed computing systems. Proceedings of the IEEE, v. 54, n. 12, p. 1901-1909, December 1966.

GRAMA, A. et al. Introduction to Parallel Computing, Second Edition. [S.l.]: Addison Wesley, 2003.

GROPP,W. et al. A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard. Parallel Computing, v.22, n.6, p.789-828, September 1996.

HENNESSY, J. L.; PATTERSON, D. Computer Architecture: A Quantitative Approach. Second Edition. San Francisco, California: Morgan Kaufmann, 1996. 760p.

LAM/MPI 2012. LAM/MPI Paralell Computing - http://www.lam-mpi.org, (Março. 2012).

MOURA e SILVA, L.; BUYYA, R. Parallel Programming Models and Paradigms. In: BUYYA, R. High Performance Cluster Computing: Programming and Applications. New Jersey: Prentice-Hall, 1999. v.2. p.4-27.

MPI CENAPD. Centro Nacional de Processamento de Alto Desempenho, http://www.cenapad.unicamp.br/servicos/treinamentos/apostilas/apostila_MPI.pdf. (Fevereiro. 2012).

MPI FORUM. MPI: A Message-passing Interface Standard. Knoxville, Tennessee: University of Tennessee, June 1995. (Technical Report, version 1.1).

NUPAIROJ, N; NI, L. Performance Evaluation of Some MPI Implementations on Workstation Clusters. In: SCALABLE PA RALLEL LIBRARIES CONFERENCE, 1994. Proceedings. IEEE Computer Society Press, October 1994. p.98-105.

PACHECO, P. Parallel Programming With MPI. [S.l.]: Morgan Kaufmann; 1st edition, 1996.

PRAMANICK, I. MPI and PVM Programming. In: BUYYA, R. High Performance Cluster Computing: Programming and Applications. New Jersey: Prentice-Hall, 1999. v.2. p.48-86.

SILBERCHATZ, A.; Galvin, P.; Gagne, G. Operating system concepts. Willey, 2008.

TANENBAUM, A. S.; Woodhull, A. Operating systems design and implementation. Prentice Hall, 2006.

UNIVERSITY OF TENNESSEE, MPI: A MESSAGE-PASSING INTERFACE STANDARD. Knoxville, Tennessee. 1995. Disponível em: <http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html #Node0>. (Março. 2012).

Capítulo 4

Composição de Serviços Web Semânticos em Nuvem

Fabrício de Oliveira Alves and Daniela Barreiro Claro

Resumo

Nos últimos anos empresas e academia têm migrado suas aplicações e documentos para as nuvens, interessados em escalabilidade, segurança e baixos custos. Muitos serviços já foram desenvolvidos para o ambiente Web, mas ainda não foram publicados em uma nuvem. Atualmente, diversos pesquisadores têm proposto ferramentas para facilitar o desenvolvimento, a utilização e principalmente a publicação destes serviços para este tipo de ambiente. O Eclipse IDE é uma das ferramentas que permitem desenvolver e publicar serviços web, porém poucos plugins foram desenvolvidos para a utilização de serviços semânticos publicados em Nuvem. Assim, este laboratório propõe introduzir os conceitos de serviços web, incluindo os aspectos semânticos e a sua relação como um serviço em nuvem (SaaS). Neste sentido, este laboratório utilizará a ferramenta OWL-S Composer 3.1, desenvolvida pelos proponentes deste laboratório, membros do grupo de pesquisa FORMAS da UFBA. O OWL-S Composer 3.1 é um plugin para a plataforma Eclipse IDE que permite a modelagem visual, descoberta e monitoramento de serviços web. Nesta ultima versão, os serviços web podem ser publicados em uma nuvem. O laboratório é dividido em duas seções: teórica e prática. Na seção teórica serão apresentadas as definições de serviços web, serviços web semânticos, composição de serviços web e os conceitos essenciais de computação em nuvem. Além disso, são apresentados os desafios concernentes à união destas tecnologias, especificamente referente à composição destes serviços publicados em nuvem. Na seção prática os participantes utilizarão a ferramenta para: i) criar um serviço web utilizando a API AXIS2; ii) criar e publicar um serviço web em nuvem (Google App Engine); iii) compor os serviços criados em nuvem;

4.1. Apresentador

Fabrício de Oliveira Alves

4.2. Objetivo do Curso

Este laboratório tem como principal objetivo capacitar os estudantes no desenvolvimento de composições de serviços web e publicação destes serviços em nuvem. O desenvolvimento de composições em nuvem permitirá ao estudante um conhecimento aprofundado em tecnologias atualizadas, tais como desenvolvimento web e na API do Google APP Engine. O estudante desenvolverá serviços semânticos, compreendendo como funciona a Web semântica e como é possível inferir semanticamente serviços similares.

Além disso, o estudante terá um material teórico referente ao domínio de composições de serviços web que facilitará a compreensão do processo de desenvolvimento de serviços semânticos.

4.3. Tratamento dado ao Tema

O tratamento dado ao tema compreende uma visão geral dos principais conceitos relacionados à área de serviços web e computação em nuvem. Dentre estas duas grandes áreas, os serviços semânticos e as composições de serviços serão enfatizados. Esta parte teórica subsidiará e norteará o desenvolvimento de serviços semânticos que são disponibilizados em nuvem. Primeiramente o estudante desenvolverá uma composição semântica de serviços e posteriormente publicará esta composição em nuvem, utilizando o Google App Engine.

4.4. Perfil desejado dos Participantes

Alunos de graduação interessados em sistemas web, computação em nuvem, serviços web, desenvolvimento de serviços na linguagem Java ou outra área de interesse relacionada ao tema.

4.5. Infraestrutura física necessária para a apresentação

Este laboratório está subdividido em uma parte teórica, de 2h30min e uma parte prática de 2h. Com o intuito que os participantes possam desenvolver suas aplicações, é necessário que o laboratório conste das seguintes ferramentas instaladas.

- Laboratório com as seguintes ferramentas instaladas: Apache Tomcat 6.0.32;
- Axis2 1.4.1;
- JDK 6 ou mais atual; Eclipse Galileo;
- Plugin OWL-S Composer 3.1[1];
- Projetor Multimídia;

4.6. Introdução

A computação em nuvem tem mudado o paradigma de armazenamento e manipulação de dados e aplicações por todos os tipos de consumidores: o que antes era feito no *Desktop* do

[1] Disponível em: http://homes.dcc.ufba.br/ dclaro/tools.html#owls3

cliente, atualmente tende a ser executado em uma nuvem [Han et al. 2009]. Baseado na disponibilização de recursos computacionais sob demanda, através da internet, a computação em nuvem tem suprido a necessidade através da escalabilidade, privacidade, segurança e, principalmente, baixos custos.

De igual forma, a utilização de serviços web tem crescido de forma significante por prover serviços padronizados, de forma interoperável, com baixos custos, integrado e baixo acoplamento. No entanto, muitas vezes utilizar apenas um serviço não atende aos requisitos do usuário, de forma que é necessário compô-los, formando um fluxo onde a saída(*output*) de um serviço é ligada à entrada(*input*) de outro.

Em um ambiente de composição automática de serviços web, a descoberta dos serviços deve analisar as entradas e saídas sob a perspectiva semântica, a fim de evitar ambiguidades. Assim, a descrição do serviço semântico utiliza linguagens como o SAWSDL [Kopecky et al. 2007], WSMO [Lausen et al. 2005] e OWLS [Martin et al. 2004].

Além disso, os desafios referentes à computação orientada a serviços devem ser revistos sob o ângulo da computação em nuvem. Estes problemas se referem principalmente a mecanismos que facilitem a descoberta, execução e composição de serviços neste ambiente.

Embora a computação em nuvem ofereça um ambiente com alta disponibilidade, a Internet continua sendo utilizada como meio de interação entre os serviços. Por ser a Internet um ambiente instável, serviços são suscetíveis a falhas, principalmente por indisponibilidade, durante o seu processo de execução. Em se tratando de aplicações críticas, este problema se agrava [Ferreira et al. 2011].

Baseado na importância destes assuntos no cenário atual, este laboratório objetiva abordar os principais conceitos de Serviços Web, suas tecnologias básicas, composição e tolerância a falhas; Computação em Nuvem, sua definição e categorização, além de abordar os problemas de utilização do ciclo de Serviço Web em um ambiente em Nuvem. Ao final do laboratório espera-se que o participante esteja familiarizado com estes conceitos e siglas e, além disso, que saiba publicar um serviço web para publicação normal e em nuvem, utilizando a ferramenta OWL-S Composer 3.1.

As próximas seções do presente capítulo estão organizadas da seguinte forma: a seção 1.7 apresenta o embasamento teórico referente aos Serviços Web; a seção 1.8 aborda a composição de serviços e a web semântica; a seção 1.9 se refere a computação em nuvem e seus conceitos. A seção 1.10 aborda o ciclo de vida de um serviço publicado em nuvem e a 1.11 aborda a evolução e arquitetura do OWL-S Composer, ferramenta utilizada neste laboratório.

4.7. Serviços Web

A Web, como projetada inicialmente, possui o objetivo principal de facilitar o compartilhamento de dados entre humanos utilizando a Internet, o que caracteriza uma comunicação humano-humano e humano-máquina. É comum a utilização da Web para compras, comunicação interpessoal, acessos a notícias, etc. Seguindo este modelo, os

dados manipulados por uma aplicação Web não podem ser utilizados diretamente por outra, visto que não satisfaz a necessidade de softwares distintos comunicarem-se [Newcomer 2002]. Desta forma, é necessário criar uma forma de permitir que esta comunicação seja feita de forma eficiente, com menor custo e sem o envolvimento humano. Os Serviços Web surgiram como opção para solucionar este problema.

Serviços Web é uma forma de disponibilização de funcionalidades de um sistema de informação na Web por meio de tecnologias padronizadas [Alonso et al. 2003]. Normalmente o serviço é identificado por uma URI, assim como qualquer outro site, diferenciando-se destes últimos no tipo de interação que o Serviço Web proporciona [Potts and Kopack 2003]. Além disso, outras vantagens conhecidas deste tipo de aplicação em relação a seus predecessores são:

- Realiza troca de documentos através de mensagens assíncronas ao invés de realizar chamadas a métodos remotos sincronamente. Isso se deve ao fato que um documento é enviado descrevendo a transação de negócio de maneira assíncrona, ou seja, o invocador não precisaria ficar bloqueado esperando a resposta. Outros serviços, que receberão como entrada o documento enviado, só precisam responder com outro documento cujo conteúdo será baseado no documento recebido, não precisando nem saber quais serviços estão envolvidos nessa operação. Essa característica promove uma das principais vantagens da utilização dos serviços Web em uma aplicação: o baixo acoplamento.

- Suporta diferentes tipos de plataforma e tecnologias que possuam suporte aos protocolos HTTP, SMTP ou FTP.

- Utiliza o Simple Object Access Protocol (SOAP) como protocolo de troca de mensagens. Esse protocolo, que funciona sobre protocolos como HTTP, SMTP ou FTP, utiliza o XML como formato de mensagem e inibe um problema que ocorria com a utilização de arquiteturas desenvolvidas anteriormente: a utilização de portas especiais de acesso. Essa vantagem aumenta a disponibilidade e a segurança dessa tecnologia, já que, firewalls, por padrão não bloqueiam mensagens SOAP visto que estas são simples requisições HTTP (ou HTTPS).

- Utiliza XML para descrição de mensagens. Essa vantagem é explicada devido ao formato do XML ser de dados auto-descrito. Ferramentas que interceptam essa mensagem, podem facilmente identificar seus atributos e entidades sem necessariamente utilizar o mesmo esquema XML da mensagem, facilitando a integração entre aplicações distribuídas distintas. Além disso, a utilização do XML permite que os serviços sejam independentes de linguagem, plataforma ou sistema operacional e garante autonomia em relação a outros serviços.

A figura 4.1 mostra todo o ciclo de vida de um serviço web, formado por seis passos importantes para as garantias de reuso, alta disponibilidade e execução de um serviço web [Fonseca et al. 2009]. Cada um destes seis passos é descrito a seguir:

Figure 4.1. Ciclo de Vida de uma Composição de Serviços Web [Fonseca et al. 2009]

- *Publicação*: Nesta fase, o serviço web é disponibilizado em forma de URI, possivelmente em um repositório, para que possa ser utilizado. É importante denotar que este repositório, que pode ser público ou privado, deve ser acessível a todos os interessados em utilizar o serviço.

- *Descoberta*: Etapa em que ocorre a varredura de repositórios pré-especificados a fim de encontrar serviços adequados à necessidade do cliente. A descoberta é um dos passos mais importantes, posto que um resultado incorreto acarretará na execução de serviços que não estarão de acordo com a requisição do cliente.

- *Seleção*: A partir da lista de serviços encontrados na etapa anterior, o cliente utiliza métricas para selecionar um ou mais serviços que se enquadrem a sua requisição.

- *Composição*: Através deste passo os serviços selecionados podem ser interligados em um fluxo de execução, onde a saída de uma operação possa ser utilizada como parâmetro para uma nova solicitação. Serviços que se enquadrem neste passo são chamados de compostos, por possuir mais de um serviço envolvido na execução. Caso contrário, são denominados atômicos.

- *Invocação*: Este passo utiliza os dados de Seleção e Composição, caso tenham ocorrido com sucesso, para executar o serviço e retornar ao cliente o resultado da operação.

- *Monitoramento*: Durante a Invocação, é necessário monitorar o serviço de forma contínua a fim de garantir que continue funcionando mesmo na presença de falhas. Geralmente são utilizadas técnicas de Tolerância a Falhas e Computação Autonômica para aumentar a disponibilidade e confiabilidade na execução destes serviços.

A Arquitetura Orientada a Serviços (SOA) surgiu em decorrência da necessidade de integrar sistemas heterogêneos, proporcionar uma melhor comunicação interna entre eles e facilitar o reuso. Em muitos trabalhos essa arquitetura é utilizada como exemplo para demonstrar as interações entre os atores em uma arquitetura utilizando serviços Web.

Como é visto na figura 4.2, existe um fluxo padrão para o processo relacionado a

SOA: (1) o provedor do serviço desenvolve a aplicação e a publica no Repositório,
(2) caso o repositório esteja disponível, o possível consumidor busca por um serviço, baseado nas Entradas, Saídas, Pré-condições e Efeitos (IOPE) e, (3) se o serviço ideal para a necessidade do cliente for encontrado, é feita a solicitação por parte do cliente, através de uma chamada ao provedor.

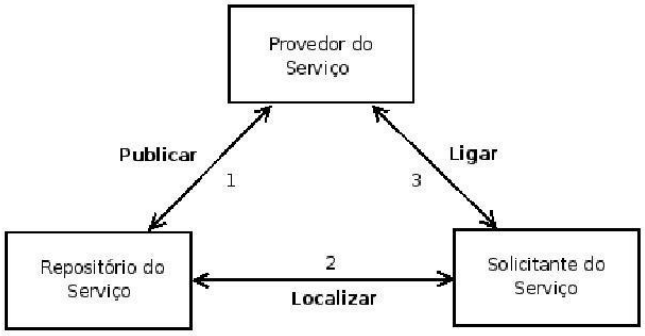


Figure 4.2. Arquitetura SOA [Fonseca et al. 2009]

4.7.1. Tecnologias Básicas

Algumas tecnologias são essenciais para o funcionamento da SOA, fornecendo padrões para troca de dados, descrição e publicação dos serviços. Duas das principais tecnologias são o *SOAP* e o *WSDL*, que serão descritos a seguir:

- **SOAP**: O *Simple Object Access Protocol* (SOAP) é o protocolo de comunicação utilizado para troca de dados entre Serviços Web. Através desta tecnologia, os dados são encapsulados em um envelope e enviados para o terminal de destino. Uma das maiores vantagens do SOAP é descrever em XML os argumentos necessários para invocação e resposta de um serviço, sem que o cliente necessite saber como estes serviços são implementados, ao contrário de soluções anteriores como DCOM e CORBA [Potts and Kopack 2003]. De acordo com [Newcomer 2002], o SOAP é talvez uma das mais significantes tecnologias dos Serviços Web, por garantir a independência de plataforma, sistema operacional e linguagem de programação. Além disso, o SOAP utiliza HTTP como forma de transporte, o que garante uma melhor comunicação e maior poder ao serviço, como, por exemplo, atravessar *firewalls*.

  Outra característica deste protocolo é a simplicidade de sua arquitetura. Uma mensagem SOAP é composta de duas partes obrigatórias: o SOAP Envelope e o SOAP Body, e uma parte opcional: o SOAP Header. O SOAP Fault deverá aparecer caso ocorra alguma erro no processamento da mensagem. Estes elementos serão melhores descritos abaixo:

  SOAP Envelope: Similar a um envelope físico, é onde estão contidos o SOAP Body e o SOAP Header. Um exemplo de SOAP Envelope é mostrado na listagem 4.1

  SOAP Header: Se estiver presente, deverá possuir informações de controle e configuração. Uma mensagem SOAP pode possuir vários Headers.

  SOAP Body: Contém o corpo da mensagem (ou payload), tanto da que está sendo

enviada (request) quanto da que está sendo recebida (response). Se seu processamento funcionou corretamente, então a response voltará com o corpo da resposta, caso contrário, o elemento response voltará com um elemento Fault, contendo a descrição do erro ocorrido.

SOAP Fault: Caso tenha ocorrido algum erro durante o processamento do request, o SOAP irá inserir um elemento Fault dentro do Body da response.

```
<?xml version=1.0 ?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  soap:encodingstyle="http://www.w3.org/2003/05/soap-e
ncoding">
  <soap:Header>
  ...
        <!-Informações de Controle->
  ...
  </ soap:Header>
  <soap:Body>
  ...
  <!- Corpo da mensagem ou Fault ->
    ...
  </ soap:Body>
</ soap:Envelope>
```

Listing 4.1. Modelo de Documento Soap

- **WSDL**: Uma aplicação que acessa um serviço Web deve conhecer como e que tipo de informação deve trocar com ele. O *Web Services Description Language* (WSDL) é um documento, baseado em XML, que descreve este tipo de serviço formalmente, provendo um modo simplificado de representar suas informações. Este descreve formalmente os metadados de um Serviço Web. Através dele, clientes podem saber quais argumentos o serviço espera receber, quais são os dados retornados como resposta e qual o protocolo de comunicação ou transporte ele suporta [Newcomer 2002]. Além de utilizar estas informações para interagir com um serviço, o cliente pode ainda utilizá-las para buscar uma aplicação que melhor preencha seus requisitos.

  Um documento WSDL é dividido logicamente em dois diferentes agrupamentos: as descrições concreta e abstrata. A descrição concreta é composta de elementos orientados à ligação física entre o cliente e o serviço. A descrição abstrata é composta de elementos orientados à descrição das capacidades do serviço (POTT; KOPACK, 2003). Os quatro elementos XML abstratos de um documento WSDL são:

  wsdl:types: É usado para indicar que um tipo WSDL está sendo declarado. Normalmente, um tipo de dados definido pelo usuário onde é composto de tipos de dados primitivos ou complexos.

  wsdl:messages: Elemento que descreve mensagens que deverão ser enviadas ou recebidas por esse serviço, contendo um conjunto de <wsdl:types>.

  wsdl:operation: Análoga a uma chamada de método em uma linguagem de programação como Java, por exemplo. Contudo somente três tipos de mensagens são permitidas em uma operação: mensagens de entrada (Input), mensagens de saída (Output), e mensagens de falha (Fault).

wsdl:portType: Conjunto de todas as operações que um serviço Web pode realizar. Os três elementos XML concretos de um documento WSDL são (POTT; KOPACK, 2003):

wsdl: service: Elemento que contém referência a todas as ports que estão contidas no documento WSDL.

wsdl:port: Elemento que contém informações de IP e porta do serviço Web que está representado no WSDL.

wsdl:binding: Esse elemento tem dois propósitos: Primeiramente, interligar elementos concretos e abstratos no documento WSDL. O segundo propósito é prover um conjunto de informações como protocolo do serviço Web. Adicionalmente, o elemento raiz do documento WSDL é o <wsdl:definitions>. Nele, é especificado o namespace do documento, também chamado de targetnamespace.

4.8. Composição de Serviços Web

Diversos serviços têm sido publicados na internet utilizando estas tecnologias e padronizações. Embora esta proliferação seja crescente, encontrar um serviço que atenda os requisitos do cliente é difícil e, muitas vezes, este não encontra um serviço simples capaz de executar a tarefa desejada, encontrando apenas combinações de serviços que podem realizá-la. Desta forma, é necessária a criação de um fluxo, de forma que a saída de um serviço seja automaticamente direcionada para a entrada de outro [Murtagh 2004]. Tal combinação é denominada *Composição de Serviços Web*. Além disso, esta estratégia pode aumentar ainda a eficiência do processo, se comparado com a execução dos serviços atômicos separadamente, pois informações úteis para executar determinado serviço podem ser reutilizadas durante a composição na solicitação de outros serviços. Dois conceitos são importantes para o contexto da Composição de Serviços Web:

- **Orquestração**: Modelo de processo que descreve a composição de serviços tendo como base um coordenador que detém o controle do fluxo de execução.
- **Coreografia**: modelo de processo que descreve a colaboração dos serviços e a interação entre as orquestrações, sem que haja um serviço que coordene tal interação.

A figura 4.3 demonstra a diferença entre Orquestração e Coreografia.

Uma composição de serviços web deve ser transparente para o cliente, sendo observada como um Serviço Web comum. Desta forma, uma composição segue o mesmo fluxo da arquitetura SOA (figura 4.2), além de possuir entradas, saídas, pré-condições e efeitos (IOPE).

De acordo com [Kumar and Mishra 2008] uma composição de serviços web pode ser automática ou manual. Na composição automática, o cliente oferece ao compositor os argumentos de entrada e o objetivo a ser alcançado e este é responsável por descobrir os serviços que podem ser compostos para satisfazer o objetivo, utilizando para tanto, algoritmos de descoberta e escolha, como planejadores de inteligência artificial [Silva and Claro 2009]. Por outro lado, o processo manual determina em tempo de projeto quais são os serviços que farão parte da composição.

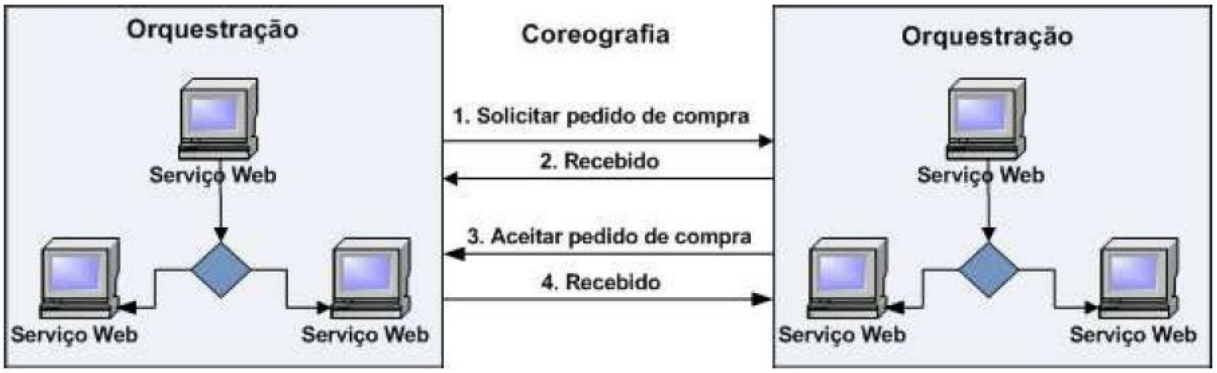


Figure 4.3. Diferença entre Orquestração e Coreografia [Michels 2009]

A composição manual é ineficiente e, considerando o tamanho da Internet atual, impraticável. Seja dado como exemplo uma aplicação que necessite converter um documento ".doc" para ".pdf" e que este serviço não existe, é necessário combinar serviços a fim de atingir ao requisito do cliente. Caso a composição seja feita de forma manual, o cliente deverá especificar o tipo intermediário (por exemplo, ".rdf"). Todavia, os serviços estão disponibilizados na Internet e são suscetíveis a falhas e, caso haja uma falha por omissão, o cliente deverá indicar, em tempo de execução, outro formato intermediário. A partir deste exemplo fica clara a ineficiência da abordagem manual. Caso o processo de escolha de um novo serviço seja automático, o usuário especifica apenas a entrada (".doc") e o objetivo (".pdf") e o compositor se encarrega de realizar a composição dos serviços necessários.

4.8.1. Composição de Serviços Web Semânticos

Uma das etapas mais importante da composição é a descoberta dos serviços participantes. A escolha incorreta de um serviço compromete todos os passos seguintes do ciclo de vida da composição. Segundo [Sheu et al. 2009], um dos maiores problemas da descoberta é assumir que os requisitos do usuário e os argumentos do serviço sempre se relacionarão de forma idêntica. De fato, é raro que ocorra tal situação no mundo real e adotar uma decisão lógica rígida resultaria em ignorar serviços que possuam argumentos similares aos requisitos informados.

Uma das soluções que resolvem este problema é unir Serviços Web com a Web Semântica. Com o aumento da quantidade de informações disponibilizadas na Web, fez-se necessário haver uma forma de permitir que tais informações fossem interpretadas também por máquinas, não somente por humanos. De acordo com [Berners-Lee et al. 2007], a Web Semântica não passa de uma extensão da Web atual, com a adição de anotações semânticas que permitem que pessoas e computadores trabalhem juntos. De forma semelhante, é possível descrever Serviços Web de forma semântica, de onde surgem os Serviços Web Semânticos (SWS).

Para exemplificar a busca de serviços utilizando SWS, considere que um cliente deseje encontrar um serviço que receba como parâmetro um argumento do tipo *Carro* e retorne um parâmetro do tipo *Preço*, simbolizando o valor da tarifa diária de aluguel do referido carro. Suponha ainda que no espaço de busca do usuário não existe um serviço com esta descrição, mas existe um serviço que recebe como parâmetro um argumento do

tipo *Veículo* e retorne um argumento do tipo *Custo*. É fácil notar que o segundo serviço satisfaz parcialmente a busca do usuário, pois possuem argumentos semelhantes. No entanto, um sistema de busca baseado apenas no WSDL não o retornaria como possível solução do problema do usuário. Por outro lado, caso estes serviços sejam SWS, um sistema de busca baseado na descrição semântica retornaria o serviço desejado baseado na similaridade semântica entre os argumentos.

Geralmente, um SWS possui, além do documento de descrição do serviço (WSDL), um documento que o descreve semanticamente, escrito em uma linguagem específica para representar as ontologias do serviço. Várias linguagens foram desenvolvidas para este fim, dentre estas é possível citar a SAWSDL[Kopecky et al. 2007], que estende a WSDL através da adição de anotações semânticas, a WSMO[Lausen et al. 2005], que provê um *framework* para descrição semântica e execução de SWS e a OWL-S[Martin et al. 2004].

A **SAWSDL** (Semantic Annotation For WSDL) é um conjunto de extensões para a WSDL. De fato, a SAWSDL isolada não provê nenhuma semântica específica. Ela permite que os documentos WSDL, que são unicamente sintáticos, possuam ponteiros para descrições semânticas. Esta linguagem descreve o serviço Web Semântico em três níveis: a *Reusable Abstract Interface* define um conjunto de operações para troca de mensagens; *Binding* define o protocolo de serialização da mensagem na rede (SOAP, por exemplo) e *Service*, que representa o serviço Web que programa uma interface e pode ser acessado através de diferentes *endpoints*. Apesar de ser o primeiro passo da W3C para a padronização das linguagens de descrição semântica [Kopecky et al. 2007], o principal ponto fraco da SAWSDL é a alteração no padrão WSDL, o que força aos provedores de serviços já publicados a alterarem a descrição destes.

A **WSMO** (Web Service Modeling Ontology) constitui o principal esforço europeu para o desenvolvimento de linguagens de descrição semântica para serviços. Ela provê um *framework* que suporta a modelagem de Serviços Web, na forma de descrição de vários aspectos semânticos relacionados a tais serviços. Para tanto, ela se baseia nos princípios de compatibilidade com padrões Web, através do uso de URIs´ (*Universal Resource Identifier*) e namespaces, utilização de ontologias para descrição de requisitos funcionais e não funcionais e a utilização de mediação entre dados e protocolos. Um documento WSMO é composto por quatro elementos: *Ontologies* - que descreve todos os outros elementos do documento, *Web Services* - que descreve semanticamente os serviços Web, incluindo as propriedades funcionais e não funcionais, *Goals* - que descreve em linguagem de alto nível os estados a serem atingidos e, por fim, *Madiators* - que garante o acoplamento e interoperabilidade entre todos os elementos arquiteturais (dados, protocolos e processos executados).

A principal vantagem da **WSMO** é a distinção clara entre orquestração e coreografia, além da diferença semântica entre objetivos do usuário e objetivos do serviço.

A **OWL-S** (Ontology Modeling Language for Services) é uma linguagem baseada na OWL *Web Ontology Language*, que, por sua vez, é uma linguagem baseada em XML para modelagem de ontologias. De acordo com [Martin et al. 2004], a **OWL-S** utiliza uma ontologia e três subontologias para definir e descrever semanticamente um serviço Web. A ontologia *service* referencia as sub-ontologias *Model*, *Profile* e *Grounding*. Cada descrição de serviço em OWL-S deve conter uma instância do *Service*. Como mostra a figura 4.4,

esta ontologia contém ainda informações relacionadas ao serviço, como limitações, pré-condições e qualidade do serviço (QoS). A ontologia *Profile* descreve as habilidades do serviço, incluindo os elementos funcionais (IOPE) e os não funcionais (QoS). Em *Model* estão descritas as informações de invocação dos serviços e troca de mensagens. Além disso, caso o serviço seja composto, esta ontologia descreve o fluxo de controle de invocação, tais como: *Sequence*, *Split*, *SplitJoin*, *AnyOrder*, *Choose*, *If-Then-Else* e *Repeat-Until*. Por último, a subontologia *Grounding* realiza o mapeamento das estruturas da subontologia *Model* com uma descrição WSDL.

Um dos pontos positivos da OWL-S é o fato de não precisar alterar a descrição WSDL de serviços disponibilizados sem anotações semânticas. Para tornar um serviço semântico, basta descrevê-lo utilizando a OWL-S e apontar este último para o WSDL referente ao serviço.

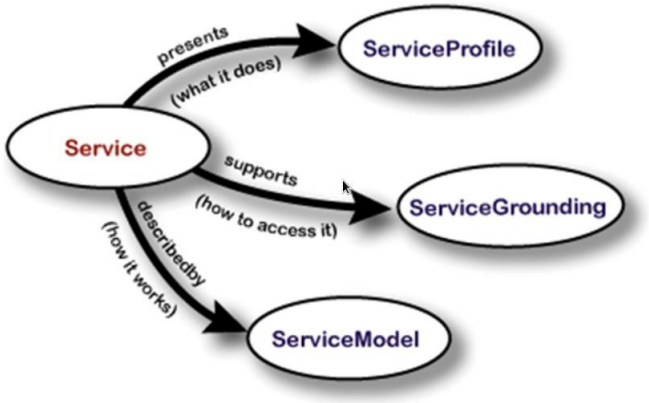


Figure 4.4. Estrutura da Ontologia Service [Martin et al. 2004]

Em [Oliveira et al. 2009] é realizado um estudo comparativo entre estas três linguagens, utilizando como métricas a estrutura da linguagem, os parâmetros IOPE (*Input, Output, Precondition and Effect*) e a descoberta, composição, mediação e execução de serviços Web. Segundo este trabalho, A WSMO é a mais completa em relação à capacidade de descrever recursos de serviços Web. No entanto, esta linguagem é também a mais complexa em relação à praticidade em utilizá-la, ou seja, o menor número de adaptações a serem realizadas para realizar as atividades do ciclo de vida de um serviço Web, devido à necessidade de um ambiente de execução próprio. Nestes quesitos, a linguagem SAWSDL configurou o oposto, sendo uma linguagem prática mas com poucos recursos, principalmente na adição de semântica e composição de serviços. Levando em consideração os quesitos de adequação dos conceitos semânticos e praticidade, a OWL-S está no meio termo entre estas duas linguagens, visto que com a utilização da OWL-S é possível descrever grande parte dos recursos de um serviço Web, com exceção da mediação de dados. Em relação à praticidade, esta linguagem não necessita de um ambiente de execução próprio, sendo mais simples que a WSMO. Esta análise é simplificada no gráfico apresentado na figura 4.5.

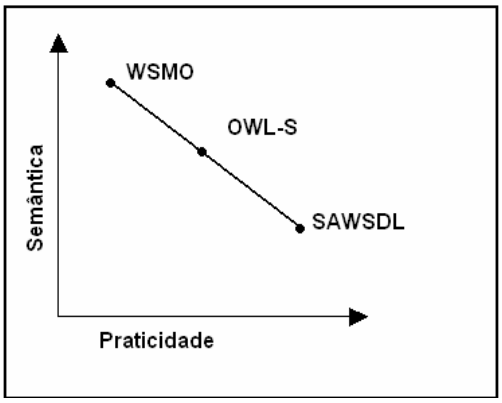


Figure 4.5. Relação entre as linguagens de descrição semântica [Oliveira et al. 2009]

4.8.2. Tolerância a Falhas na Composição de Serviços Web

Serviços Web são normalmente publicados em ambientes complexos, heterogêneos e instáveis, tais como a Internet, e estão suscetíveis a falhas por indisponibilidade, sobrecarga por excesso de chamadas, dentre outros [Ferreira et al. 2011]. No cenário atual, a falha de um serviço pode ser muito custosa em termos de preço, visto que este serviço pode ser vital na execução de atividades de grande importância no ambiente do consumidor. No contexto de composição de serviços web, estas falhas possuem uma probabilidade maior de ocorrer, visto que a execução com sucesso de uma composição geralmente depende da execução com sucesso de cada um dos serviços atômicos. Assim, faz-se necessário que haja mecanismos que garantam a execução correta da composição, seja minimizando o tempo em que o serviço falho fique indisponível, ou substituindo-o por um serviço similar, num processo de autogerência.

Tolerância a falhas se refere à propriedade de uma aplicação evitar que um serviço falhe na presença de erros [Avižienis et al. 2004]. Ao detectar um erro, a aplicação tolerante a falhas deve definir estratégias para que o consumidor do serviço não perceba quaisquer alterações no fluxo normal da operação. As estratégias a serem definidas geralmente dependem de um dos quatro tipos de falhas que podem ocorrer:

- Por Parada (*Crash*): O servidor para abruptamente enquanto executa normalmente.
- Por Omissão: O servidor não consegue responder às requisições enviadas. Pode ser dividida em dois tipos, *Omissão de Recebimento*, quando o servidor não consegue receber as mensagens que chegam; ou *Omissão por envio* quando o servidor não consegue enviar mensagens;
- De Temporarização: A requisição realizada para o servidor leva muito tempo para obter resposta.
- De Resposta: O valor retornado pela operação está incorreto.

Segundo [Ferreira et al. 2011], ao encontrar um erro durante a execução de um serviço, um conjunto de passos podem ser utilizados para garantir que uma

composição seja executada ainda que um de seus componentes falhe. Tais passos são executados na ordem em que são apresentados abaixo:

- **Reexecutar (*Retry*)**: Tenta reexecutar o serviço falho novamente.

- **Substituição**

  Por Uma Réplica (*ReplaceByEqual*): Procura nos servidores pré-definidos por um serviço que seja exatamente igual ao serviço falho, ou seja, que possua a mesma assinatura. O novo serviço encontrado será executado e seu resultado será utilizado no fluxo da composição.

  Por um serviço equivalente (*ReplaceByEquivalent*): Encontra um serviço que seja semanticamente similar ao que falhou. Caso seja encontrado um novo serviço, este será executado e o resultado será utilizado no fluxo da composição.

- **Saltar (*SkipOptionalService*)**: No fluxo de composição, alguns serviços podem ser não essenciais para o resultado desejado, o que os torna facultativos. Esta estratégia visa identificar se o serviço que falhou é obrigatório ou facultativo no fluxo da composição e, caso seja facultativo, continuar a execução do fluxo sem o resultado obtido pelo serviço. Um serviço é considerado facultativo se não gera saídas ou se as saídas geradas não estão interligadas com alguma entrada de outro serviço.

4.9. Computação em Nuvem

A utilização dos serviços discutidos até agora têm crescido de forma significante nos últimos anos. De fato, hoje a grande maioria dos negócios ou principais atividades usam, ou dependem de alguma forma, de TI ou serviços de TI [Vouk 2008]. Tais serviços devem possuir alta disponibilidade, segurança, privacidade, escalabilidade e, possivelmente, baixo custo.

No entanto, manter um servidor dedicado que atenda a estas necessidades é um trabalho difícil que envolve recursos de alto valor, visto que servidores quebram, requerem a compra, instalação, upgrade e desinstalação de softwares caros, além de se tornarem defasados, lentos e carregados de vírus [Blacharski and Landis 2010][Hayes 2008]. Estudos apontam que a maioria dos *Data Centers* utilizam 99% do seu poder computacional em 10% do tempo. No restante do tempo a capacidade computacional é subutilizada o que gera custos desnecessários de backup, redundância, bem como de montar servidores com grande poder de processamento, capazes de manipular grandes transferências de dados em picos que, como visto, ocorrem de forma infrequente.

Visando solucionar estes problemas, no ano de 2007 foi anunciado o conceito de *Computação em Nuvem* como um novo modelo de publicar serviços utilizando o poder computacional do *Grid Computing*, a versatilidade e escalabilidade da Virtualização e as características de cobrança e utilização sob demanda da *Utility Computing* e a solução baseada em Internet dos *Application Service Provider* (ASP). [Zhang et al. 2010].

A definição para Computação em Nuvem é dada pelo *National Institute of Standards and Technology* (NIST)[2], pois aborda os principais conceitos acerca de Cloud Computing:

> "Computação em nuvem é um modelo que possibilita acesso via Internet, de forma conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis (tais como redes, servidores, bancos de dados, aplicações e serviços) que podem ser rapidamente fornecidos e publicados com mínimo esforço gerencial ou interação com o provedor de serviços"[3].

De modo geral, esta definição inclui conceitos chaves da computação em nuvem, relacionados a arquitetura, segurança e estratégias de publicação. Os principais elementos que são articulados nessa definição são:

- **Serviço Sob Demanda**: Este elemento foi herdado da *Utility Computing* e diz respeito a tornar o custo dos serviços computacionais semelhantes à de indústrias mais consolidadas na área de serviços, como eletricidade e petróleo. Um consumidor com uma necessidade particular pode utilizar mais recursos computacionais (tais como memória, poder de processamento, uso de softwares ou capacidade de armazenamento) sem que haja a interferência humana com os administradores do serviço. Além disso, o cliente paga apenas pelo montante de recursos que utilizar e o provedor da cloud deve conseguir medir o uso de tais recursos por cada um dos clientes [Dillon et al. 2010].

- **Acesso via Internet**: Como demonstrado na Figura 4.12 os recursos computacionais são disponibilizados através da Internet e acessados por aplicações de diferentes plataformas - computadores pessoais, *PDAs*, *celulares*, *laptops*, *et cetera*.

- **Agrupamento de Recursos**: Os recursos computacionais do provedor de serviço são agrupados para atender a vários consumidores, através da multilocação e virtualização. Tal agrupamento permite que os recursos computacionais sejam transparentes ao cliente, o que dá a este a impressão de ter uma quantidade infinita de recursos a sua disposição, além de permitir que cada ambiente seja customizado ao locatário do serviço.

[2] Disponível em: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
[3] Tradução do autor

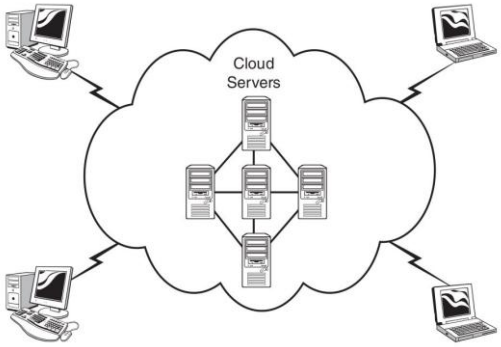


Figure 4.6. Cloud e a conexão com a Internet [Miller 2009]

A depender de suas finalidades e restrições de acesso, uma nuvem pode ser classificada, através do seu modelo de desenvolvimento, em Privada, Pública ou Híbrida [Zhang et al. 2010]. As nuvens *Privadas* são operadas dentro de uma organização e gerenciadas pela própria ou por terceiros contrados para este fim. As principais motivações para construir nuvens privadas são a maximização e otimização dos serviços disponíveis apenas para seus donos, além de oferecer um maior controle de desempenho, disponibilidade e segurança. Nas nuvens *Públicas* o processo ocorre de forma mais aberta. Os provedores dos serviços os disponbilizam para acesso público, com garantias de baixo ou nenhum investimento inicial em arquitetura. Os benefícios destes dois tipos podem ser unidos em uma nuvem *Híbrida*, o que adiciona flexibilidade na provisão do serviço.

4.9.1. Categorias de Serviço em Nuvem

A computação em nuvem é divida em categorias de serviços distintas, onde cada uma das camadas arquiteturais podem ser providas como um serviço sob demanda. No entanto, conforme demonstrado na Figura 4.7, nuvens oferecem serviços que podem ser agrupados em apenas três diferentes categorias [Mell and Grance 2009]: IaaS, PaaS e SaaS.

- **Infrastructure as a Service (IaaS)**

  Como visto na Figura 4.7, a categoria *Infrastructure as a Service* compreende as camadas de Hardware e Infraestrutura da arquitetura em nuvem. Este modelo está apto a prover uma estrutura física (processador, memória, armazenamento, dentre outros) em forma de serviço, onde o cliente possa rodar, por exemplo, um sistema operacional. No entanto, esta estrutura física deve ser provida de forma que obedeça à propriedade da computação em nuvem que indica que todo serviço deve ser sob demanda. A virtualização é muito utilizada em IaaS para resolver este problema.

  Ao invés de recursos físicos, o provedor do serviço disponibiliza para os clientes máquinas virtuais que podem escalar conforme a sua necessidade, além de permitir que multilocatários utilizem o mesmo recurso físico com total isolamento um dos outros.

- **Platform as a Service (PaaS)**

  Platform as a Service compreende a camada de mesmo nome na arquitetura. PaaS oferece uma framework de desenvolvimento que suporta todo o ciclo de vida do software, o que permite que os clientes desenvolvam serviços e aplicações em nuvem

[Dillon et al. 2010]. Como toda plataforma, o PaaS oferece um invólucro de recursos de software e o cliente desenvolve um produto que se encaixe perfeitamente neste invólucro. Como pontos positivos pode-se citar a diminuição do custo de engenharia de software, tempo de publicação e riscos, além de aumentar os níveis de segurança e requerer menos conhecimento técnico por parte dos desenvolvedores [Blacharski and Landis 2010]. Por outro lado, as aplicações desenvolvidas possuem menos customização e, muitas vezes, podem ter seus escopos reduzidos para se adaptarem ao modelo de desenvolvimento proposto pela plataforma. Um exemplo é o *Google App Engine*[4]. Esta plataforma permite a integração como o Eclipse e o desenvolvimento e publicação de aplicações web. No entanto, o desenvolvedor é obrigado a escolher entre duas linguagens de programação, além de não poder utilizar todas as bibliotecas presentes nestas linguagens.

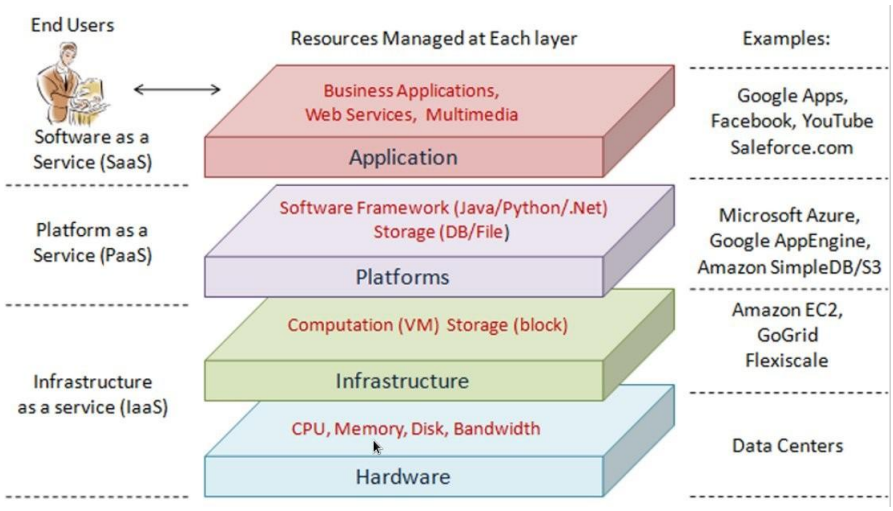


Figure 4.7. Arquitetura e Categorias de serviços na computação em nuvem [Zhang et al. 2010]

- **Software as a Service(SaaS)**

É a categoria que se refere à camada de mais alto nível. Entende-se por SaaS aplicações desenvolvidas para serem executadas em nuvem por multilocatários, onde tais locatários não podem interferir na plataforma ou infra estrutura. Esta forma elimina a necessidade de instalação e execução do software no computador pessoal do cliente. Como é o modelo mais visível, SaaS dita o crescimento da computação em nuvem e é o tipo mais utilizado pelos consumidores [Blacharski and Landis 2010]. Além de incorporar os já citados pontos positivos da computação em nuvem, a grande vantagem desse modelo é a fácil manutenção dos aplicativos. Como apenas uma instância do aplicativo é disponibilizada, as etapas de correção de erros, refinamentos de código, atualizações de segurança e *upgrade* de funcionalidades é feita apenas uma vez, o que torna o processo mais rápido e menos dispendioso.

Serviços web podem ser utilizados no contexto de computação em nuvem para criação, publicação, utilização e interconexão entre serviços. Além disso, um serviço web, por ser um tipo de software especial, pode ser disponibilizado em formato de SaaS reunindo os benefícios tanto da computação orientada a serviços, quanto da computação em nuvem. Segundo [Erl 2011], a utilização de computação em nuvem

[4] http://appengine.google.com

com SOA resulta em menores investimentos em desenvolvimento, aumento de agilidade e capacidade de resposta e confiabilidade dos serviços publicados.

4.10. Ciclo de Vida de Um Serviço Disponibilizado em Nuvem

Assim como um serviço web comum, um serviço disponibilizado em nuvem possui o ciclo de vida descrito na figura 4.1, composto por *Publicação*, *Descoberta*, *Seleção*, *Composição, Invocação* e *Monitoramento*. Os passos de Seleção e Invocação independem de onde o serviço é disponibilizado. Os demais passos são brevemente modificados a fim de suprir as necessidades do ambiente em nuvem.

4.10.1. Publicação

Atualmente, com o sucesso da computação em nuvem, existem vários provedores deste tipo de serviço. No entanto, a forma de disponibilizar serviços Web é desenvolvê-lo sobre uma PaaS, a qual é a categoria menos difundida. Ainda assim, existem bons exemplos de PaaS presentes no mercado:

- Microsoft Azure (http://www.windowsazure.com/) - Possui ferramentas, bibliotecas e API's próprias para executar aplicações desenvolvidas em .NET;
- Force.com (http://www.force.com/) - Especializada em aplicações do tipo CRM e voltadas para o gerenciamento de clientes, empresas, etc.
- Heroku (http://www.heroku.com) - Possui ferramentas, bibliotecas e API's próprias para execução de aplicações desenvolvidas em Ruby on Rails;
- CloudBees (http://www.cloudbees.com) - Possui invólucro para desenvolvimento de aplicações Web em Java;
- Google App Engine (http://appengine.google.com) - Possui invólucro para desenvolvimento de aplicações Web em Java e Python;

Um dos maiores pontos fracos da publicação de serviços em ambiente em nuvem é que o desenvolvimento de aplicações deste tipo não é padronizado e depende intrinsecamente da PaaS utilizada. Um exemplo disto é que aplicações desenvolvidas para a plataforma Google App Engine utilizando a linguagem Java não são aptas a executarem na plataforma CloudBees. Um outro ponto negativo é que muitas das PaaS presentes atualmente no mercado - tais como Microsoft Azure, Force.com, Heroku e CloudBees - não possuem suporte para SOAP e, quando possuem, não permitem a utilização de ferramentas que auxiliem no desenvolvimento e publicação deste tipo de serviço, como Axis [Apache 2007] e JAX-WS [Java.net 2011], caso do Google App Engine[Google 2008]. A Amazon EC2 [Amazon 2007] é uma plataforma que possui suporte a SOAP, através dos protocolos SOAP e RestFul, mas não permite o uso livre da ferramenta para testes acadêmicos, o que dificulta a utilização desta plataforma. É importante deixar claro que a disponibilização de um serviço em uma PaaS não faz deste serviço um SaaS. Para que este seja um SaaS, é necessário que o mesmo possua características intríssecas da computação em nuvem, por exemplo a utilização de recursos sob demanda, como visto na seção 4.9.

4.10.2. Criação de um Serviço Web para Publicação no Google App Engine

O Google App Engine (GAE) é uma PaaS voltada para a execução de aplicativos Web desenvolvidos nas linguagens Python, Java (ou qualquer outra linguagem cujo interpretador rode sobre a máquina virtual Java). Apesar de não permitir a utilização dos pacotes Axis e JAX-WS, o que dificulta o desenvolvimento de serviços Web, esta plataforma possui suporte para a SOA, através do protocolo SOAP. Outro ponto positivo é que o GAE oferece um pacote de serviços grátis para publicação de aplicações, sendo a cobrança de taxas feitas apenas nos momentos em que a aplicação ultrapassa a quantidade de uso de recursos computacionais limitados pelo pacote grátis. Além disso, a plataforma disponibiliza o Google Plugin Para Eclipse [Google 2011], que oferece os módulos, pacotes e classes neessários para o desenvolvimento de aplicações em conformidade com o GAE.

O primeiro passo é gerar a estrutura básica do serviço, o *Servlet* que responderá pelas requisições, os módulos externos do GAE que interagirão com o servidor Web, além do SDK (Software Development Kit) do Google. Todos estes artefatos são gerados no momento de criação do projeto, utilizando o *plugin*. Após a codificação normal do serviço em Java, devem ser feitas as anotações *java.soap.WebService* e *java.soap.WebMethod*, como visto na listagem 4.2 que indicam qual é o serviço e quais são os métodos disponibilizados, respectivamente. Caso o JAX-WS fosse suportado no GAE, o desenvolvimento do serviço seria finalizado neste passo. No entanto, como não existe tal suporte, é necessário realizar as seguintes etapas manualmente:

```
package com.example;
import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService //Anotação que indica qual é o serviço
Public class ClimaInfo{

    @WebMethod   //Anotação   que   indica   qual   método   é
disponibilizado através do serviço
    Public String comoEstaOTempo(int temperatura){
        if(temperatura>30){
            return "Quente";
        }
        Else if(temperatura<15){
            return "Frio";
        }
        else{
            return "Agradavel";
        }
    }
}
```

Listing 4.2. Anotações que Indicam o Serviço Web e os Métodos disponibilizados

1. *Gerar o WSDL*: a geração do WSDL deve ser feito de forma manual. Todavia, as anotações feitas no código permitem a utilização do aplicativo *wsgen*, nativo do Java, que baseia-se nestas anotações para criar o descritor WSDL além de gerar os

POJOS. *POJOS* são as classes responsáveis por traduzir as informações vindas à partir do protocolo SOAP, em XML, para a linguagem Java. São criados dois POJOS para cada método do serviço: o primeiro trata os dados de Requisição para o serviço (número e tipos de parâmetros, por exemplo) e o segundo trata os dados referentes à Resposta do serviço.

2. *Criar Adaptador entre POJO e Serviço*: Uma classe deve ser gerada para realizar a adaptação entre o *POJO* e o serviço desenvolvido. Em outras palavras, este adaptador deverá receber um POJO de requisição, realizar a chamada à classe princpal utilizando os argumentos recebidos e retornar uma resposta no formato de um *POJO* de resposta.

3. *Criar Manuseador de Requisição*: Ao receber a requisição em XML, a aplicação deve ser apta a reconhecer qual o *POJO* de requisição deve ser criado, quais os parâmetros recebidos, etc. Todavia, esta requisição é encapsulada através do SOAP, sendo necessário que o XML seja traduzido para que os POJOS corretos sejam chamados. Logo, uma classe manuseadora deste XML é criada, para fazer a ligação entre a requisição e os *POJOS*.

4. *Criar o Roteamento para a Requisição SOAP*: Por fim, é necessário direcionar o fluxo da requisição SOAP para o manuseador criado. Assim, o servlet gerado automaticamente pelo Google *plugin* deve receber uma função que leia o cabeçalho da requisição, reconheça a chamada SOAP e direcione para o manuseador devido.

Um diagrama de classes para o desenvolvimento de um serviço Web para publicação no Google App Engine é visto na figura 4.8. O teste do serviço criado pode ser feito através do próprio plugin, que possui um servidor Web para testes embarcado. A publicação para a nuvem é feita também utilizando o próprio plugin. Faz-se necessário deixar claro que a aplicação só será publicada se estiver em conformidade com a plataforma Google App Engine.

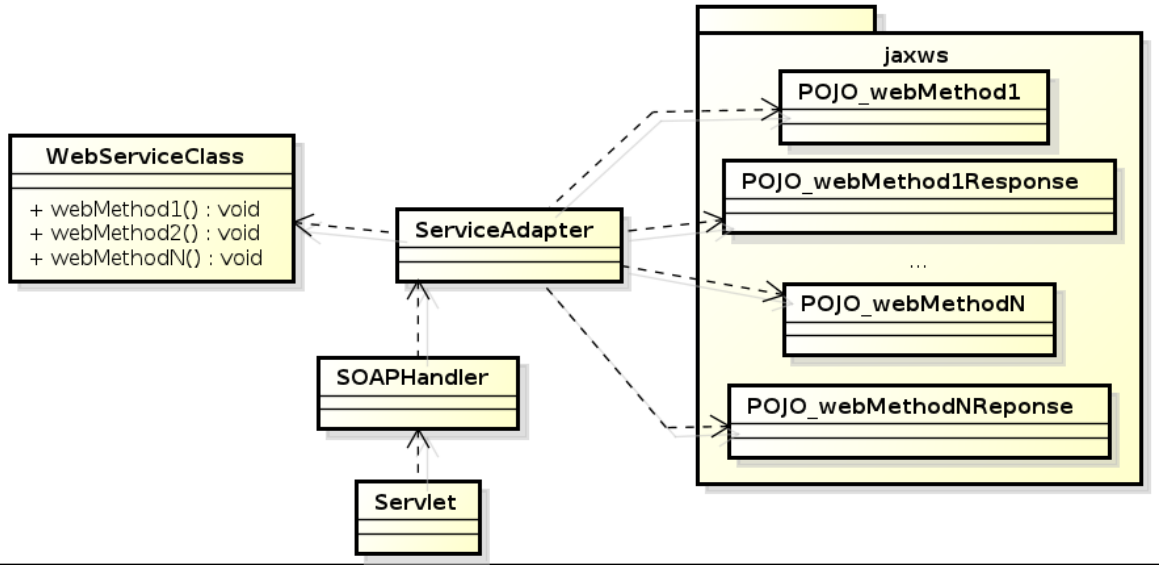


Figure 4.8. Diagrama de Classes do Desenvolvimento de Serviços para o Google App Engine

4.10.3. Descoberta de Serviços em Nuvem

Para realizar a descoberta de possíveis serviços participantes da composição, é necessário que a ferramenta saiba em que nuvem os arquivos owls estão publicados, visto que no processo de monitoramento e recuperação de serviços falhos esta informação é muito importante.

O fato do sistema de descoberta precisar saber a URI da nuvem onde o serviço está hospedado, muito se deve ao não consenso na academia em utilizar repositórios de serviços, além do fato de grande parte destes repositórios não permitiram a adição de anotações semânticas [Ferreira et al. 2011].

Assim é necessária a inclusão de um arquivo de especificação de publicação de serviços, descrito em XML, que tem o objetivo principal de especificar quais serviços estão disponibilizados em uma determinada nuvem. A especificação não é extensiva, visto que cada serviço já possui seus próprios descritores OWL-S e WSDL. Apesar de possuir o inconveniente de gerar um artefato extra, este documento tem a vantagem de simplificar a tarefa do usuário do serviço, que precisará conhecer apenas a URI do arquivo. Além disso, é possível também saber qual a PaaS utilizada, informação que pode ser útil em algum momento.

A existência de um arquivo de especificação e publicação de serviços limita a Descoberta de Serviços em Nuvem a um escopo mais reduzido, como por exemplo, escolher um número X de nuvens participantes da composição onde serão buscados um serviço semanticamente similar a um serviço falho.

Na listagem 4.3 pode-se ver o esquema do documento aceito pelo *OWL-S Composer 3.1*.

```
<xs:elementname="cloudProvider">
  <xs:complexType>
          <xs:sequence>
          <xs:elementname="service"minOccours="1"maxOccours="unbounded"><xs:co
  mplexType>
                  <xs:sequence>
                          <xs:elementname="name"type="xs:string"/>
                  </xs:sequence>
                  <xs:attributename="descriptorURI"type="xs:string"/>
          </xs:complexType>
          </xs:sequence>
          <xs:attributename="URI"type="xs:string"/>
          <xs:attributename="platform"type="xs:string"use="optional"/>
  </xs:complexType>
</xs:element>
```

Listing 4.3. Esquema do Arquivo de Especificação de Publicação

O principal elemento do documento é o *cloudProvider*, que representa o provedor de serviços, identificado pelo atributo *URI*. Opcionalmente, o provedor do serviço pode possuir também o atributo *platform*, que indica qual o PaaS onde o serviço foi disponibilizado. Apenas uma instância do *cloudProvider* é permitida. Dentro do elemento principal, estão listados cada um dos serviços, identificados pelo atributo *descriptorName*, que indica o filename do arquivo owl que descreve o serviço, e pelo elemento *name*, referente ao nome do serviço.

A listagem 4.4 mostra um exemplo de arquivo de especificação, contendo os serviços *celsiusToFarenheit* e *farenheitToKelvin*, ambos publicados na plataforma *GoogleAppEngine*.

```
<cloudProviderURI="http://meutcc.com/services"platform="GoogleAppEngine">
      <service descriptorName="celsiusToFarenheit.owl">
            <serviceName>celsiusToFarenheit</serviceName>
      </service>
      <service descriptorName="farenheitToKelvin.owl">
            <serviceName>farenheitToKelvin</serviceName>
      </service>
</cloudProvider>
```

Listing 4.4. Exemplo de expecificação de publicação com dois serviços

4.10.4. Composição de Serviços em nuvem

A interconexão entre nuvens distintas tem sido um ponto bastante abordado atualmente. Esta integração recebe denominações, dependendo de sua finalidade. Uma aplicação *intercloud* ou *Multi Cloud* utiliza serviços disponibilizados em nuvens diferentes através de serviços web ou compartilhamento de máquinas virtuais. Já a aplicação *Cross Cloud* é um tipo especial de *intercloud* que visa a escalabilidade e portabilidade dos serviços, através da federação de nuvens distintas. Neste modelo, o usuário que utiliza máquinas virtuais em uma nuvem (*Home Cloud*) pode requerer máquinas virtuais que estejam em uma outra nuvem (*Foreign Cloud*) dentro da mesma federação.

Através da provisão de composição de serviços web disponibilizados em nuvem, é possível alcançar também a interconexão entre nuvens. Desta forma, é criada uma nova categoria de serviços, chamada **Composition as a Service (CaaS)**[Michlmayr et al. 2007], que incorpora a integração e abrangência provida pela composição de serviços web, com a economia e escalabilidade do modelo de computação em nuvem.

4.10.5. Monitoração e Recuperação de Serviços em Nuvem

Até o momento, os métodos de composição de serviços assumem que todos os serviços web selecionados para a composição estão publicados no mesmo repositório. No entanto, no caso da CaaS, é comum que provedores de serviços publiquem seus serviços em diferentes PaaS. Em um ambiente de composição manual de serviços, o usuário pode escolher os serviços de maneira que achar mais conveniente, por outro lado, em um ambiente de composição automática, o compositor deve levar em consideração a utilização do menor número de nuvens possível, visto que a comunicação entre serviços web entre nuvens distintas pode ser custosa, tanto em relação à taxa a ser paga, quanto ao tempo de execução e interconexão entre as nuvens [Zou et al. 2010]. Da mesma forma, no contexto de sistemas tolerantes a falha, caso ocorra uma falha na execução de um serviço, é preferível que o serviço escolhido para substituí-lo esteja na mesma nuvem, ou em uma nuvem já utilizada na composição, a fim de obter tais benefícios.

Assim, é necessário estender a lista de estratégias de recuperação já apresentadas na seção 4.8.2, a fim de garantir que um número pequeno de nuvens distintas sejam utilizadas caso ocorram falhas na execução da composição, conforme é apresentado abaixo:

- **Reexecutar (*Retry*)**: Tenta reexecutar o serviço falho novamente.

- **Substituição**

  Por Uma Réplica Nas Nuvens Participantes (*ReplaceByEqualInCloud*): Efetua uma busca dentro das nuvens participantes da composição visando selecionar um serviço que seja uma réplica, ou seja possui parâmetros de entrada e saída com o mesmo tipo do serviço falho.

  Por Uma Réplica (*ReplaceByEqual*): Efetua uma busca dentro de todos os diretórios, inclusive as nuvens pesquisadas no passo anterior. Escolhe uma réplica do serviço falho, exceto o serviço já utilizado no passo anterior.

  Por um Equivalente Nas Nuvens Participantes (*ReplaceByEquivalentIn-Cloud*): Tem como objetivo selecionar serviços semanticamente similares para substituir o que foi identificado como falho. A busca é feita dentro das nuvens utilizadas na composição, utilizando a ferramenta OWL-S Discovery [Amorim et al. 2011], para identificação em tempo de execução destes serviços.

  Por um Equivalente (*ReplaceByEquivalent*): Realiza a mesma tarefa do passo anterior, ou seja, realiza a busca de serviços semanticamente similares para substituir o serviço que falhou. No entanto, o espaço de busca é aumentado para todos os diretórios pré-estabelecidos. Assim como no passo *ReplaceByEqual*, a escolha executa o serviço escolhido no passo anterior.

- **Saltar (*SkipOptionalService*)**: No fluxo de composição, alguns serviços podem ser não essenciais para o resultado desejado, o que os torna facultativos. Esta estratégia visa identificar se o serviço que falhou é obrigatório ou facultativo no fluxo da composição e, caso seja facultativo, continuar a execução do fluxo sem o resultado obtido pelo serviço. Um serviço é considerado facultativo se não gera saídas ou se as saídas geradas não estão interligadas com alguma entrada de outro serviço.

É importante deixar claro que a ordem de execução das estratégias de recuperação é seguinte: 1) Reexecuta o serviço falho. Caso o serviço execute sem falhas, finaliza o processo de recuperação. Caso contrário utiliza a estratégia 2. 2) Realiza uma busca por uma Réplica nas Nuvens participantes. Caso o serviço execute sem falhas, finalizar processo de recuperação. Caso contrário, utiliza a estratégia 3. E assim sucessivamente.

4.11. OWL-S Composer 3.1

A ferramenta utilizada neste minicurso para a visualização e fixação dos conceitos de Serviços Web e Computação em Nuvem é o OWL-S Composer 3.1. Esta ferramenta foi desenvolvida pelo grupo FORMAS[5] a fim de suportar todo o ciclo de vida de um Serviço Web disponibilizado em Nuvem.

4.11.1. Histórico

A evolução da *ferramenta* desde sua primeira versão até a 3.0 é apresentada na figura 4.9.

O *OWL-S Composer 1.0* [Fonseca et al. 2009] é um *plugin* para a plataforma

[5] http://www.wiki.dcc.ufba.br/FORMAS

Eclipse[6] que visa a composição visual de SWS. O forte desta ferramenta é a integração com *plugins* do ambiente Eclipse, o que permite a criação e manutenção de serviços de forma mais intuitiva para o usuário, por serem amplamente utilizadas no cenário atual. Para tanto, a interface visual do *OWL-S Composer 1.0* é composta por um editor de diagramas, onde a composição é modelada visualmente, através de polígonos e retas que, respectvimante, abstraem os serviços e mapeam as entradas e saídas do serviços compostos. Na Figura 4.10 esta interface é visualizada, na qual foi gerada uma composição do tipo *Sequence* que liga a saída do serviço *getMatriculaAluno* com uma das entradas do serviço *matricularAluno*. Os três tipos distintos de estrutura de controle presentes no *plugin* são: *Sequence*, *Split* e *AnyOrder*.

Para a construção da ferramenta, o pacote *Eclipse Modeling Tools* foi muito im portante, visto que o mesmo disponibiliza *plugins* que foram utilizados na construção do *OWL-S Composer*:

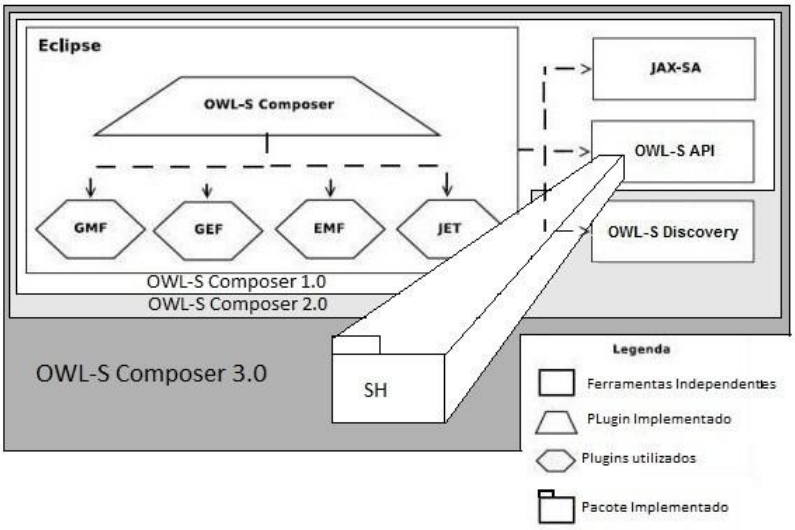


Figure 4.9. Evolução da Arquitetura do OWL-S Composer até a versão 3.1

- **EMF**: O EMF (Eclipse Modeling Framework Project) [EMF 2009] permite a geração de código à partir do diagrama criado ou, ao contrário, a geração do modelo à partir do código. Por exemplo, é possível através do **EMF** gerar código Java à partir de diagramas UML. No *OWL-S Composer 1.0* o *EMF* é utilizado para gerar o documento OWL-S da composição, baseado na versão 1.1 desta linguagem.

- **GEF**: O GEF (Graphical Editing Framework)*framework* [GEF 2009] torna possível a criação de editores de diagrama à partir do modelo da aplicação. Este *plugin* utiliza a arquitetura *Model-View-Controller* (MVC), o que facilita a integração do modelo com seus elementos gráficos. A arquitetura utilizada ajuda ainda a aumentar a legibilidade e manutenabilidade do código dos diagramas. No *OWL-S Composer 1.0* este *plugin* fornece os mecanismos necessários para a construção da composição de serviços através de elementos gráficos.

- **GMF**: A intermediação entre o **EMF** e o **GEF** é possibilitada através do GMF

[6] http://www.eclipse.org

(Graphical Modeling Framework)[GMF 2009]. Para tanto, o **GMF** facilita o trabalho do desenvolvedor, ao permitir que o mapeamento entre diagrama e código a ser gerado seja feito de forma visual. O mapeamento entre a modelagem e os elementos do documento OWL-S no *OWL-S Composer 1.0* é feita com o **GMF**.

- **JET**: O JET (Java Emitter Templates) [JET 2009] gera código Java à partir de *templates* desenvolvidos à partir de *Java Server Pages* (JSP). Através do **JET** o *OWL-S Composer 1.0* converte o diagrama de composição em código Java.

Foram utilizadas também as seguintes ferramentas independentes da plataforma Eclipse: a **OWL-S API** [MINDSWAP 2007] fornece o necessário para o tratamento de arquivos *OWL-S*, tais como leitura e escrita, além da execução de serviços disponibilizados através desta linguagem. Esta API foi escolhida por ser a mais utilizada no cenário atual; a **JAX-SA** [BABIK 2008] permite a anotação semântica de Serviços Web e foi utilizada na conversão de documentos WSDL para OWL-S. No *OWL-S Composer 1.0* a conversão entre tais documentos é opcional, visto que a ferramenta aceita também que o usuário importe arquivos *OWL-S* previamente gerados para a composição de serviços.

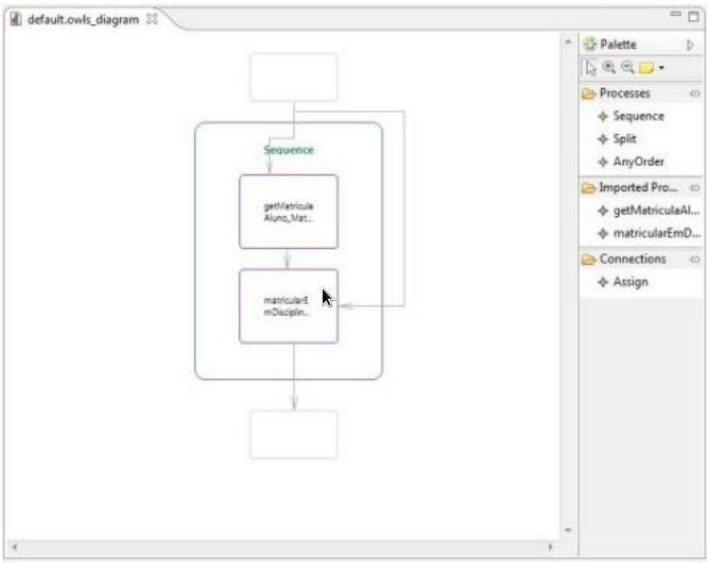


Figure 4.10. Exemplo de Diagrama Criado pelo OWL-S Composer 1.0 [Fonseca et al. 2009]

O *OWL-S Composer 1.0* foi então evoluído para o *OWL-S Composer 2.0* [Sena et al. 2010], com o intuito principal de descobrir e gerar composições semanticamente similares a que foi gerada manualmente. Para tanto, o autor utiliza a ferramenta **OWL-S Discovery**, que trata-se de um algoritmo híbrido com duas etapas: Semântico Funcional e Semântico Descritiva. Na etapa Semântico Funcional é feita a comparação entre as entradas e saídas da requisição com os parâmetros dos serviços presentes no repositório. A correspondência entre tais parâmetros é definida através da ontologia de domínio especificada para cada parâmetro e o grau de similaridade entre eles é feito seguindo as classificações *exact*, *plugin*, *subsume*, *sibling* e *fail*, baseadas na ontologia. A etapa Semântico Descritiva faz uma consulta a um dicionário de sinônimos construído pelo desenvolvedor e fornecido ao sistema de comparação entre classes. Esta etapa baseia-se no fato de que duas classes serão tão semanticamente similares quanto os seus vizinhos

estruturais e divide as classes em quatro grupos: *Ancestrais*, *Irmãos*, *Filhos Diretos* e *Folhas*. O algoritmo de similaridade básica então retorna 1.0 (um) se as classes são iguais, 0.0 (zero) se elas são diferentes ou
0.5 se as classes são similares, de acordo como dicionário de sinônimos. A similaridade entre as duas classes é então definida através da soma ponderada dos algoritmos de similaridade estrutural (Semântico Funcional) e similaridade básica (Semântico Descritiva).

A versão *OWL-S Composer 3.0* [Ferreira et al. 2011] integra mecanismos de auto-cura à versão anterior do *plugin*, visando aumentar a confiabilidade na execução de composições de serviços. O objetivo é cumprido através do módulo *sh*, implementado na *OWL-S API*, que se destina ao monitoramento e execução da composição. Para garantir a tolerância a falhas, caso haja um erro na execução, este módulo aplica as estratégias de recuperação apresentadas na seção 4.8.2.

Por fim, a última versão (3.1) do *plugin* adapta a versão 3.0 para suportar também a publicação, composição e monitoramento de serviços web disponibilizados em nuvem.

4.11.2. Arquitetura do OWL-S Composer 3.1

A arquitetura do OWL-S Composer teve apenas uma alteração em relação à sua versão anterior, já vista na figura 4.9. Dentro do módulo principal do plugin, o *owls*, foi criado o módulo *cloud* que visa embarcar todas as classes e pacotes que resolvam problemas diretamente ligados a computação em nuvem. Nesta versão do projeto, esta classe é responsável pela leitura e tradução do documento de especificação de publicação. No futuro, este módulo pode ser estendido para atuar de forma específica com cada provedor de nuvem diferente. A figura 4.11 demonstra o diagrama de módulos do plugin e onde ocorreu a inserção do submódulo *cloud*.

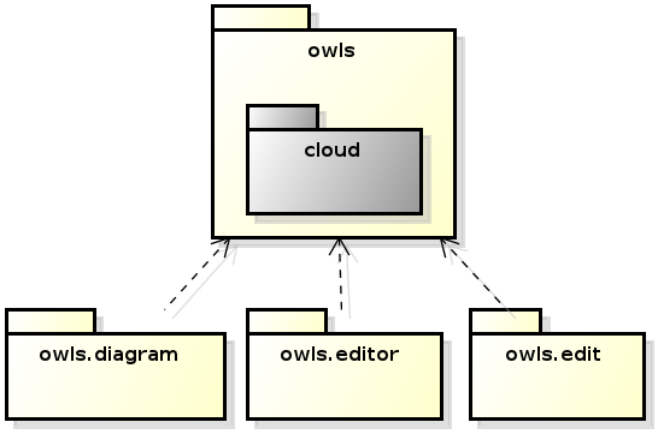


Figure 4.11. Diagrama de módulos do OWL-S Composer e a inserção do submódulo *cloud*

A etapa de recuperação de falhas requereu mudanças na estrutura do módulo *sh* implementado no *OWL-S Composer 3.0*. O módulo *sh* segue a estrutura MDR (Moni toramento, Diagnóstico e Recuperação) [Ferreira et al. 2011], visto na figura 4.12, que gerencia a execução de um serviço web. Assim, existem três fases: *Monitoramento*, que identifica a ocorrência de falhas do elemento gerenciado. Caso uma falha seja identificada, a fase de *Diagnóstico* a avalia e seleciona a estratégia de recuperação correspondente que é, por fim, executada pela fase de *Recuperação*. A implementação

desta estrutura no *OWL-S Composer 3.0* foi feita de forma automática e dinâmica, sem a necessidade do usuário modificar os arquivos OWL-S ou criar arquivos auxiliares. No entanto, a versão desenvolvida neste trabalho, necessita de um arquivo auxiliar que descreve os serviços publicados em um provedor, para prover o controle da recuperação em serviços disponibilizados em nuvem. Na figura 4.13, que mostra a arquitetura do OWL-S API, o módulo em destaque é o *sh*, que foi desenvolvido na versão anterior e alterado na presente versão deste trabalho.

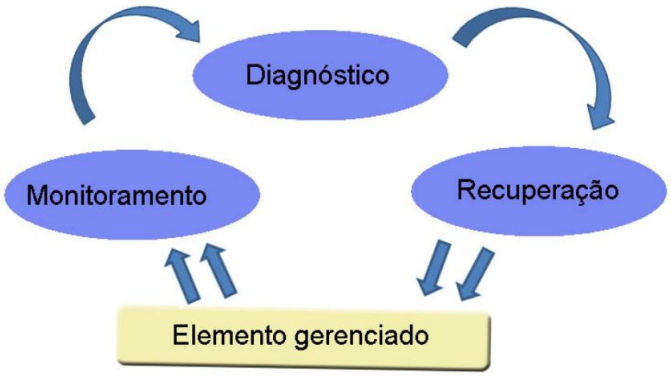


Figure 4.12. Estrutura MDR [Ferreira et al. 2011]

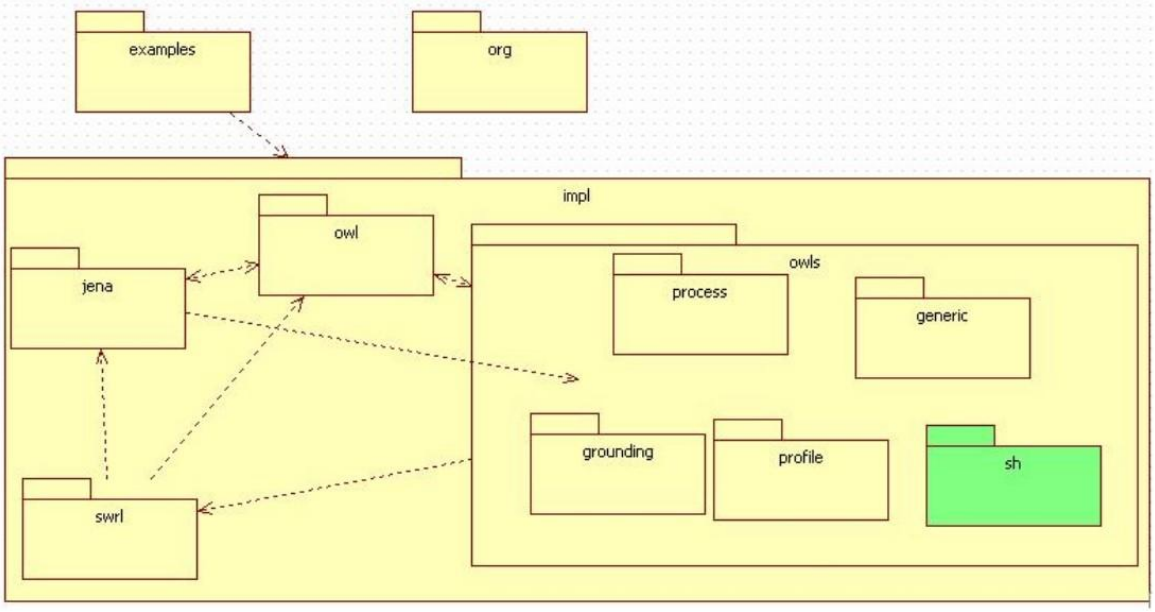


Figure 4.13. Módulo alterado na OWL-S API

4.11.3. Experimentos no OWL-S Composer

Foram realizados experimentos que testaram as funcionalidades, publicação, execução e tolerância a falhas de um serviço, tanto os disponibilizados em nuvem, quanto os que são

disponibilizados em servidores comuns. Para tanto, foi gerado um serviço composto, denominado *NameAutoPriceTechnology*, que recebe modelo de carro como entrada através do parâmetro *Model*. Este serviço retorna o preço e a tecnologia referentes ao modelo do carro passado como parâmetro de entrada. Tais retornos são chamados de *Price* e *Technology*. Além disso, é utilizado um objeto do tipo *Auto* para intermediar a comunicação entre os serviços atômicos participantes da composição.

Cada um dos parâmetros citados acima é especificado através de uma ontologia. É importante relembrar que a utilização de ontologias é importante para a fase de descoberta de serviços semanticamente similares, que é utilizada na substituição de serviços falhos for um equivalente. Cada uma destas ontologias são especificadas a seguir:

Na Figura 4.14 está representada a hierarquia da classe *Auto*. *Auto* possui *Car* como classe filha e *WheeledVehicle* como pai. *Model* é uma propriedade que pertence à classe *DesignedThing* relacionada a *Auto*.

Na Figura 4.15 está representada a hierarquia da classe *Technology*. *Technology* é um *Thing* e têm *InformationTechnology* e *ComputingTechnology* como filhos.

Na Figura 4.16 está representada a hierarquia da classe *Price*. A classe *Price* é filha de *UntangibleObjects* e *MaxPrice*, *RecommendedPrice*, *TaxFreePrice* e *TaxedPrice* são seus filhos.

O tipo *Model* é caracterizado por ser um tipo que não possui outras classes na sua hierarquia.

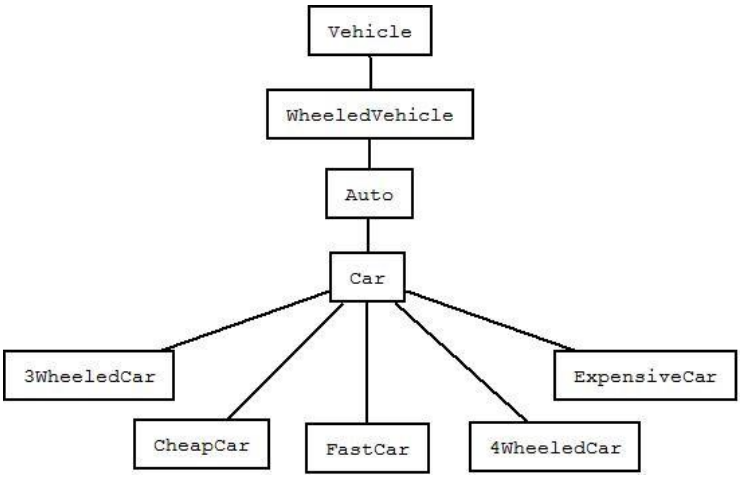


Figure 4.14. Ontologia *Auto*

O serviço *NameAutoPriceTechnology* foi modelado utilizando os controladores de fluxo *Any-Order* e *Sequence*, além disso, é composto por serviços atômicos. A estrutura da *sequence* é composta pelo serviço atômico *getNameCar_NameCar*, que possui como entrada um *Model* e como saída um *Auto* e uma estrutura de *Any-Order*. A estrutura *sequence* obriga que o fluxo siga de acordo como seja modelado, diferente do *Any-Order* que permite que os serviços sejam executados em qualquer ordem, não havendo dependência entre parâmetros. A estrutura *Any-Order* modelada contêm o serviço *get-CarPrice_CarPrice* que possui como entrada um *Auto* e como saída *Price* e o serviço *getAutoTechnology_AutoTechnology* que também possui um *Auto* como entrada, mas como saída tem *Technology*. Na Figura 4.17 é ilustrada essa composição modelada no

OWL-S Composer 3.1. Nos experimentos realizados para testar as composições em nuvem foram utilizadas três disposições diferentes: 1) Todos os serviços disponibilizados na mesma nuvem; 2) Serviços disponibilizados em nuvens distintas; 3) Parte dos serviços disponibilizados em nuvem e parte dos serviços publicados localmente.

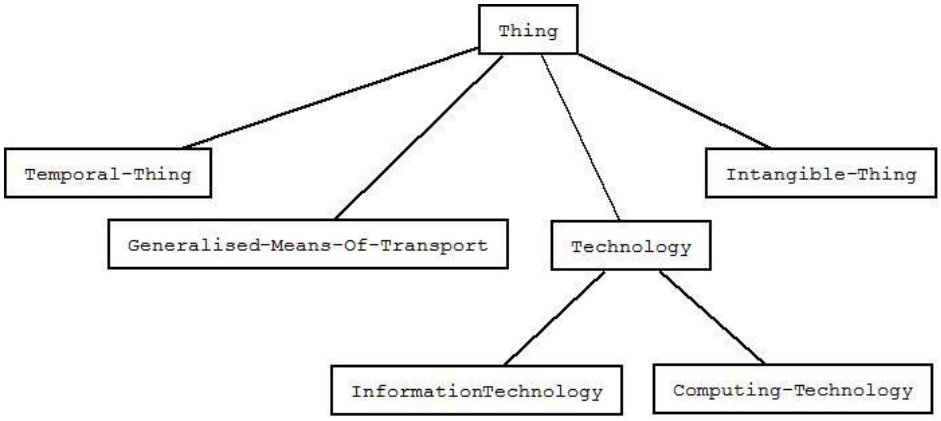


Figure 4.15. Ontologia *Technology*

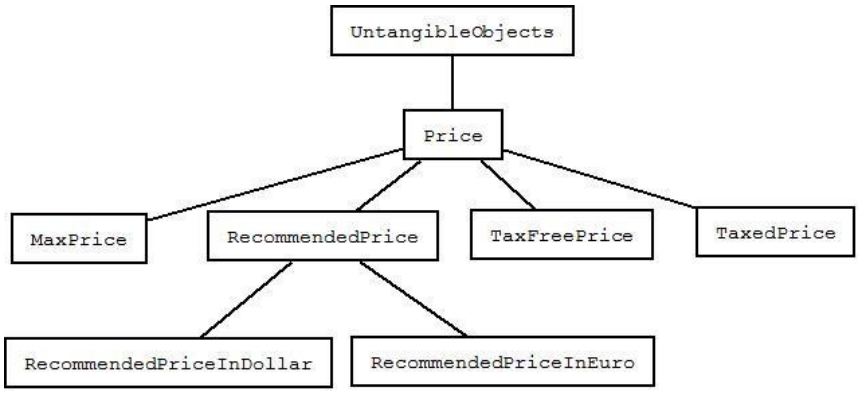


Figure 4.16. Ontologia *Price*

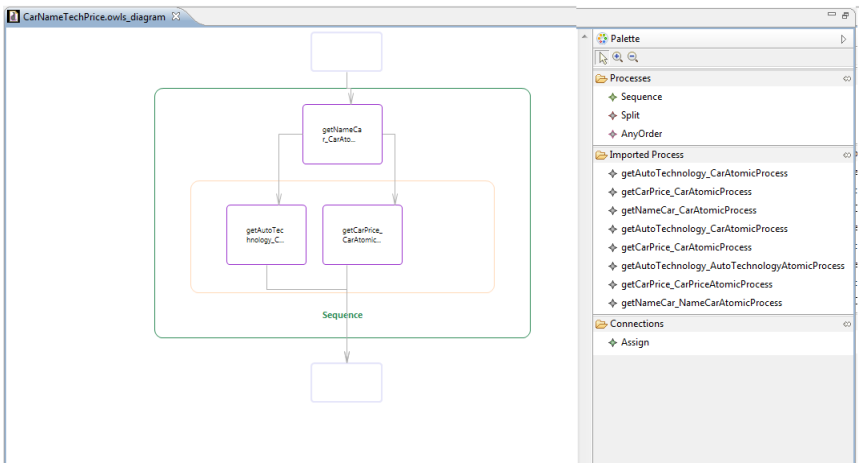


Figure 4.17. Representação da composição *NameAutoPriceTechnology* através do *OWL-S Composer 3.0*

A sequência de experimentos realizados foi a seguinte:

- **Execução de composição sem falhas**: Para testar o sistema de execução de serviços, a composição foi executada sem adição de falhas;

- **Mecanismo de Reexecução**: Um *loop* infinito foi introduzido no serviço *getNameCar*. Assim, ao executar este serviço é levantada uma falha por TIMEOUT e há a tentativa de reexecutá-lo.

- **Mecanismo de Substituição por Réplica**: Neste teste, o loop infinito foi novamente introduzido e, além disso, uma réplica do serviço *getNameCar* correta foi publicada em outra nuvem. Desta forma, existe a tentativa de execução e, como existe a falha, de reexecução. Como o serviço não foi consertado, o mecanismo de tolerância a falhas utilizou o sistema *ReplaceByEqual* e encontrou a réplica com sucesso. a réplica existente em outra nuvem.

- **Mecanismo de Substituição por Equivalente**: A réplica correta do serviço foi retirado e em seu lugar foi adicionado o serviço *getAutoName*, semanticamente similar ao *getNameCar*. Desta forma, os mesmos passos do teste anterior foram repetidos, mas ao tentar achar uma réplica, esta não será encontrada. O mecanismo de execução realizará então a busca por um serviço similar e o encontrará em outra nuvem.

- **Mecanismo de Saltar**: Para realizar este teste, a saída do serviço *getNameCar* foi retirada do conjunto de saídas da composição. Assim, ao falhar em todas as opções anteriores, o serviço foi verificado como desnecessário e retirado do fluxo de composição.

À partir dos testes executados acima, foi possível comparar a ferramenta com outras publicadas anteriormente, a saber *SHIWS*[Denaro et al. 2007], *OWL-S Composer 2.0* [Sena et al. 2010]. Baseado em [Chafle et al. 2007], [Fonseca et al. 2009] e [Sena et al. 2010], essas ferramentas foram comparadas segundo os requisitos Funcionalidade, Interface, Usabilidade, Integração, Legibilidade e Grau de similaridade. Além disso foram utilizados também os requisitos Execução e Monitoramento que estão atrelados ao critério Funcionalidade. A tabela 4.1 mostra o resultado desta comparação para o *OWL-S Composer 3.0*. É importante ressaltar que, embora a comparação tenha sido feita com a versão anterior do plugin, a comparação continua válida para a versão 3.1, visto que não houveram grandes alterações estruturais.

Além disso, foram testados também o desempenho e consumo de memória da ferramenta OWL-S Composer 3.1 em relação à sua versão anterior. Pelo motivo de ter que baixar todos os arquivos *.owl* de todas as nuvens envolvidas na composição, o desempenho e consumo de memória foram 10% a 20% maiores na versão 3.1.

Table 4.1. Tabela comparativa no contexto ferramenta [Ferreira et al. 2011]

	SHIWS	*OWL-S Composer 2.0*	*OWL-S Composer 3.0*
1. Funcionalidade	Parcial	Parcial	Parcial
1.1 Execução	Sim	Não	Sim

1.2 Monitoramento	Parcial	Não	Sim
2. Interface	Sim	Sim	Sim
3. Usabilidade	Não	Sim	Sim
4. Integração	Sim	Sim	Sim
5. Legibilidade	Não	Sim	Sim
6. Grau de similaridade	Não	Sim	Sim

4.12. Referências

Alonso, G., Casati, F., Kuno, H., and Machiraju, V. (2003). *Web Services - Concepts, Architectures and Applications*. Springer, 1 edition. Amazon (2007). Amazon ec2. http://aws.amazon.com/pt/ec2/.

Amorim, R., Claro, D. B., Lopes, D., Albers, P., and Andrade, A. (2011). Improving web service discovery by a functional and structural approach. In *IEEE ICWS 2011 - The 9th International Conference of Web Services*.

Apache (2007). Web services axis. http://axis.apache.org/axis/.

Avižienis, A., Laprie, J., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, 01(1):11–33.

BABIK, M. (2008). Jax-sa. http://sourceforge.net/projects/jax-sa/.

Berners-Lee, T., Hendler, J., and Lassila (2007). The semantic web. *Scientific American*.

Blacharski, D. and Landis, C. (2010). *Cloud Computing Made Easy*. Lulu.com.

Chafle, G., Das, G., Dasgupta, K., Kumar, A., Mittal, S., Mukherjea, S., and Srivastava,

B. (2007). An integrated development enviroment for web service composition. *IEEE International Conference on Web Services*, pages 839–847.

Denaro, G., Pezze, M., and Tosi, D. (2007). Shiws: A self-healing integrator for web services. In *ICSE COMPANION '07: Companion to the proceedings of the 29th International Conference on Software Engineering*, pages 55–56, Washington, DC, USA. IEEE Computer Society.

Dillon, T., Wu, C., and Chang, E. (2010). Cloud computing: Issues and challenges. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 27 –33.

EMF (2009). Eclipse modeling framework project. http://www.eclipse.org/modeling/emf/.

Erl, T. (2011). Soa, cloud computing & semantic web technology: Understanding how they can work together. http://www.kzoinnovations.com/afei/ppt/DayTwoKeyPMErl.pdf.

Ferreira, M., Claro, D., and Lopes, D. (2011). Integração do mecanismo de self-healing na execução da composição de sistemas de informação através dos serviços web semânticos. *VII Simpósio Brasileiro de Sistemas de Informação*.

Fonseca, A. A., Claro, D. B., and Lopes, D. C. P. (2009). Gerenciando o desenvolvimento de uma composição de serviços web semânticos através do owl-s composer. In *Proceedings of the 5th Brazilian Symposium of Information Systems (SBSI2009)*.

GEF (2009). Graphical editing framework. http://www.eclipse.org/gef/.

GMF (2009). Graphical modeling framework. http://www.eclipse.org/modeling/gmf/.

Google (2008). Google app engine. https://appengine.google.com/.

Google (2011). Google plugin for eclipse. http://code.google.com/intl/pt-BR/eclipse/.

Han, L., Xu, Z., and Yao, Q. (2009). An approach to web service composition based on service-ontology. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference on*, volume 2, pages 173 –177.

Hayes, B. (2008). Cloud computing. *Communications Of The ACM*, 51:9 – 11.

Java.net (2011). Jax-ws. http://jax-ws.java.net/.

JET (2009). Java emitter templates. http://www.eclipse.org/modeling/m2t/?project=jet.

Kopecky, J., Vitvar, T., Bournez, C., and Farrell, J. (2007). Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6):60–67.

Kumar, S. and Mishra, R. B. (2008). Trs: System for recommending semantic web service composition approaches. *World Academy of Science, Engineering and Technology 47 2008*.

Lausen, H., Polleres, A., and Roman, D. (2005). Web service modeling ontology (wsmo).

W3C Member Submission. Último acesso em 20 de novembro de 2011.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2004). Owl-s semantic markup for web services.

Mell, P. and Grance, T. (2009). The nist definition of cloud computing. *National Institute of Standards and Technology*, 53.

Michels, P. H. (2009). Coreografia de serviços web. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Universidade Federal de Santa Catarina. Orientador: Renato Fileto.

Michlmayr, A., RosenbergVienna, F., Platzer, C., Treiber, M., and Dustdar, S. (2007). Towards recovering the broken soa triangle: a software engineering perspective. In *Proceedings of Brazilian Symposium on Software Engineering*, pages 22–28.

Miller, M. (2009). *Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online*. Que Publishing.

MINDSWAP (2007). Maryland information and network dynamics lab semantic web agents project. http://www.mindswap.org/2004/owl-s/services.shtml.

Murtagh, D. (2004). Automated web service composition. Technical report. Dissertação de Mestrado - University of Dublin.

Newcomer, E. (2002). *Understanding Web Services: XML, WSDL, SOAP and UDDI*. Number ISBN 0201750813. Addison-Wesley Longman Publishing.

Oliveira, D., Menegazzo, C., and Claro, D. B. (2009). Uma análise conceitual das lingagenes semânticas de serviços web focando nas composições: Comparação entre owl-s, wsmo e sawsdl. In *IADIS Conferência Ibero-Americana WWW/Internet (CIAWI 2009)*.

Potts, S. and Kopack, M. (2003). *Sams Teach Yourself Web Services in 24 Hours*. Number ISBN 0672325152. Sams Publishing.

Sena, V. A., Claro, D. B., Amorim, R., and Lopes, D. (2010). Similaridade semântica na composição de sistemas de informação através dos serviços web. In *VI Simpósio Brasileiro de Sistemas de Informação (SBSI 2010)*.

Sheu, P.-Y., Wang, S., Wang, Q., Hao, K., and Paul, R. (2009). Semantic computing, cloud computing, and semantic search engine. In *Semantic Computing, 2009. ICSC '09. IEEE International Conference on*, pages 654 –657.

Silva, M. V. A. and Claro, D. B. (2009). Transplan: Uma ferramenta para mapear e planejar serviços web. *8th International Information and Telecommunication Technologies Symposium*.

Vouk, M. A. (2008). Cloud computing - issues, research and implementations. *Journal of Computing and Information Technology*, pages 235 – 246.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1:7–18. 10.1007/s13174-010-0007-6.

Zou, G., Chen, Y., Xiang, Y., Huang, R., and Xu, Y. (2010). Ai planning and combinatorial optimization for web service composition in cloud computing. In *Proceedings of the International Conference on Cloud Computing and Virtualization*.

Minicursos

Capítulo 5

Introdução ao Gerenciamento Ágil de Projetos com o SCRUM

Fernando Kenji Kamei[1, 2], Alexandre Marcos Lins de Vasconcelos[2], Fabio Queda Bueno da Silva[2]

[1]Instituto Federal do Sertão Pernambucano (IF Sertão-PE), Ouricuri Estrada do Tamboril, s/n, CEP: 56200-000. Ouricuri-PE

[2]Centro de Informática - Universidade Federal de Pernambuco (CIn/UFPE) Av. Jornalista Anibal Fernandes, s/n, Cidade Universitária, CEP: 50740-560. Recife-PE

Abstract

The search for excellence and continuous improvement of software quality, along with highly competitive market that require robust and complex solutions, has made that new software development methodologies, such as the Agile ones, are proposed in order to deal with more dynamic solutions to develop software. Based on Agile Methodologies, came the Agile Project Management with the iterative and incremental project life cycle, which accepts changes during the project as an inseparable part of its development process in order to always meet the customer needs. As an example of this approach there is the Scrum, a non-prescriptive framework, designed for projects where there is great instability of the requirements and which cannot predict all the changes that will occur from the beginning to the end of the project. To perform the work in projects on unstable environments, where the ability of adaptability and rapid changes are needed, Scrum provides tools and project management practices that offer greater visibility at all stages of the project, facilitating the necessary adjustments to focus towards the desired goal. This chapter aims to present the theory and key concepts of Scrum, as well as information about its use in practice that may be useful for anyone involved with software development projects.

Resumo

A busca pela excelência e melhoria contínua da qualidade de software, juntamente com a alta competitividade dos mercados que demandam por soluções robustas e complexas, tem feito com que novas metodologias de desenvolvimento de software sejam propostas, como as Metodologias Ágeis, que possuem soluções mais dinâmicas para desenvolver softwares. Com base nas Metodologias Ágeis, surgiu o Gerenciamento Ágil de Projetos, com o ciclo de vida do projeto iterativo e incremental, e que aceita as mudanças durante o projeto como parte inseparável do seu processo, de modo a atender sempre as necessidades do cliente. Como exemplo dessa abordagem existe o Scrum, um framework não prescritivo, concebido para projetos onde existe uma grande instabilidade dos requisitos e que não conseguem prever tudo o que irá ocorrer do início ao fim do projeto sem que ocorram mudanças. Para realizar o trabalho com projetos de ambientes instáveis, onde a capacidade de adaptabilidade e respostas rápidas as mudanças são necessárias, o Scrum provê ferramentas e práticas de gerenciamento de projeto que oferecem maior visibilidade em todas as etapas do projeto, facilitando realizar os ajustes necessários para focar na direção da meta desejada. Este capítulo tem por objetivo apresentar a teoria e os principais conceitos sobre o Scrum, além de apresentar informações sobre o seu uso na prática que podem ser úteis para qualquer envolvido com projetos de desenvolvimento de software.

5.1. Introdução

Em razão do rápido crescimento da indústria de software, juntamente com o aumento da demanda por soluções cada vez mais robustas e, ainda, com requisitos mutáveis, a busca pela excelência e melhoria contínua da qualidade de software tem aumentado ao longo do tempo [Chow e Cao, 2008]. Por isso que segundo Texeira e Delamaro (2008), diversos métodos, técnicas e ferramentas têm sido propostas e utilizadas, visando o aumento da produtividade no desenvolvimento de software.

Sommerville (2007) afirma que existe uma estreita relação entre a qualidade de um processo de desenvolvimento de desenvolvimento de software e a qualidade dos produtos desenvolvidos por meio deste processo. Esse processo pode ser apoiado por uma metodologia, abordagem disciplinada para o desenvolvimento de software com o objetivo de tornar o processo mais previsível e eficiente. No entanto, nem todas as metodologias têm sido adequadas, pois existe uma alta taxa de insucesso no desenvolvimento de software atribuídas principalmente à complexidade de algumas metodologias tradicionais que realizam um levantamento completo dos requisitos, e ainda pela dificuldade destas de lidarem com as mudanças trazidas pela atual dinâmica e evolução do ambiente de negócios [Ho, 2006 et al; Ferreira e Cohen, 2008].

A alta taxa de insucesso dos projetos de software, a complexidade e dinâmica do mercado global, somada a mudança de comportamento dos clientes, fornecedores e concorrentes, o mercado têm exigido mudança comportamental das empresas de software para serem capazes de responder as mudanças exigidas pelo mercado, a fim de garantir os objetivos de prazo, custo e qualidade. Por isso que fez surgirem novas metodologias de desenvolvimento de software com o intuito de encontrar melhores definições de processos visando o aumento na qualidade de software e a satisfação do cliente. Assim surgiram as Metodologias Ágeis, que possuem dinâmica de processos flexíveis, adaptativos, e que

aceitam as mudanças como parte inseparável do seu processo de desenvolvimento. Baseado nas práticas ágeis surgiu o Gerenciamento Ágil de Projeto, que tem como premissa o desenvolvimento iterativo e incremental, onde os processos são adaptativos e que abraçam as mudanças.

Como exemplo de Gerenciamento Ágil, existe o *framework Scrum*, criado por Hirotaka Takeuchi e Ikujiro Nonaka, como um modelo de gerenciamento de projetos em indústrias automobilísticas. Na indústria de software, o *Scrum* foi inserido por Ken Schwaber, que descreveu o *Scrum* como um processo prescritivo, e que uma de suas premissas é não acreditar na possibilidade de se prever no início do projeto todas as necessidades do mesmo [Schwaber, 2004].

Este trabalho tem por objetivo apresentar a teoria e os principais conceitos sobre o *Scrum*, além da motivação para o seu surgimento. Com base na experiência dos autores deste capítulo com o uso do *Scrum*, serão apresentadas dicas e informações a respeito do seu na prática, que podem ser úteis para qualquer envolvido com projetos de desenvolvimento de software.

Este capítulo está organizado da seguinte maneira:

- Na Seção 5.2 é apresentado o referencial teórico, que constitui as bases conceituais para o bom entendimento de todo o capítulo. Inicialmente é apresentado o surgimento das Metodologias Ágeis explicando sua origem, seus princípios e características.
- Na Seção 5.3 é apresentado o framework Scrum, objetivo principal deste trabalho. De modo a explicar o seu surgimento, seus pilares, as suas principais características, e como funciona o ciclo de vida deste framework de gerenciamento ágil de projetos.
- Na Seção 5.4 são apresentadas algumas dicas que podem ser úteis para iniciantes a utilizarem o Scrum na prática em seus projetos.
- Na Seção 5.5 são apresentados alguns indicadores de qualidade que podem ser obtidos a partir do uso do Scrum como framework de gerenciamento de projetos.
- Na Seção 5.6 são apresentadas algumas ferramentas computacionais que podem ser utilizadas para auxiliar no gerenciamento de projetos com o *Scrum.*
- Na Seção 5.7 são apresentadas algumas dificuldades e desafios que podem ser enfrentados pelos projetos que fazem o uso do *Scrum,* com as respectivas alternativas de soluções baseadas na experiência dos autores.

5.2. Revisão da Literatura

Esta seção tem por objetivo apresentar as principais teorias e conceitos sobre as Metodologias Ágeis, de modo que tais conceitos sejam suficientes para um bom entendimento e aprofundamento sobre o *Scrum.* Estes conceitos e teorias são resultantes de uma revisão da literatura, e experiência prática com o *Scrum.*

A Seção 5.2.1 apresenta uma revisão sobre as Metodologias Ágeis, apresentando a motivação para o seu surgimento e as suas principais características.

5.2.1. Metodologias Ágeis

As Metodologias Ágeis são abordagens contemporâneas para criação de software com base na colaboração com o cliente, trabalho em equipe, desenvolvimento iterativo e incremental, e com respostas às mudanças [Rico, Sayani e Sone, 2009]. Tais métodos têm emergido nessa última década, se tornando uma alternativa ao desenvolvimento de software tradicional [Ktata e Lévesque, 2009], objetivando eliminar a dependência em realizar um extenso planejamento inicial e documentação de todos os requisitos do sistema [Ferreira e Cohen, 2008], com propostas que dão ênfase a flexibilidade, comunicação informal e código funcionando [Capiluppi *et al*., 2007].

Cockburn e Highsmith (2001), afirmam que as metodologias ágeis enfatizam talentos e habilidades inerentes aos indivíduos, moldando o processo as pessoas e equipes específicas. Portanto, para uma metodologia se enquadrar nas metodologias ágeis, segundo [Abrahamsson *et al*., 2002], deve utilizar o desenvolvimento iterativo e incremental, valorizar a colaboração e comunicação entre cliente e toda a equipe, ser adaptativa, com capacidade de responder às mudanças [Williams *et al*., 2007].

Baseados nas metodologias ágeis, diversos métodos têm sido propostos, tais como: *Adaptative Software Development*, *Crystal*, *Dynamic Systems Development*, *eXtreme Programming (XP)*, *Feature Driven Development*, e *Scrum* [Boehm, 2006].

Nas próximas subseções é apresentado como surgiram as Metodologias Ágeis, e quais são os seus princípios.

5.2.1.1. Origem das Metodologias Ágeis

A definição oficial de Desenvolvimento Ágil de Software foi definida sob forma de um manifesto, publicado em fevereiro de 2001 por um grupo de 17 especialistas em desenvolvimento de software, que se reuniram com o objetivo de propor maneiras melhores de desenvolver softwares. Este grupo foi denominado de Aliança Ágil.

Após dois dias de debates, o grupo não chegou ao consenso em propor uma única metodologia, porém resultou na identificação de 12 princípios e valores que as metodologias ágeis deveriam seguir. Publicaram também o Manifesto Ágil [Alliance, 2001], como descrito a seguir:

> "Estamos evidenciando maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:
> **Indivíduos e interações** mais que processos e ferramentas;
> **Software em funcionamento** mais que documentação abrangente;
> **Colaboração com o cliente** mais que negociação de contratos;
> **Responder a mudanças** mais que seguir um plano.
> Ou seja, mesmo tendo valor os itens à direita, nós valorizamos mais os itens à esquerda." [Alliance, 2001]

5.2.1.2. Princípios do Manifesto Ágil

A Aliança Ágil publicou 12 princípios que são seguidos pelo Manifesto Ágil, descritos a seguir:

> "Nossa maior prioridade é satisfazer o cliente
> através da entrega contínua e adiantada
> de software com valor agregado.
>
> Mudanças nos requisitos são bem-vindas,
> mesmo tardiamente no desenvolvimento.
> Processos ágeis tiram vantagem das
> mudanças visando vantagem competitiva para o cliente.
>
> Entregar freqüentemente software funcionando,
> de poucas semanas a poucos meses,
> com preferência à menor escala de tempo.
>
> Pessoas de negócio e desenvolvedores devem trabalhar
> diariamente em conjunto por todo o projeto.
>
> Construa projetos em torno de indivíduos motivados.
> Dê a eles o ambiente e o suporte necessário
> e confie neles para fazer o trabalho.
>
> O método mais eficiente e eficaz de transmitir
> informações para e entre uma equipe de desenvolvimento
> é através de conversa face a face.
>
> Software funcionando é a medida primária de progresso.
>
> Os processos ágeis promovem desenvolvimento
> sustentável. Os patrocinadores, desenvolvedores e
> usuários devem ser capazes de manter um ritmo
> constante indefinidamente.
>
> Contínua atenção à excelência técnica e bom design
> aumenta a agilidade.
>
> Simplicidade--a arte de maximizar a quantidade de
> trabalho não realizado--é essencial.
>
> As melhores arquiteturas, requisitos e designs
> emergem de equipes auto-organizáveis.
>
> Em intervalos regulares, a equipe reflete sobre como
> se tornar mais eficaz e então refina e ajusta seu
> comportamento de acordo." [Alliance, 2001]

5.3. *Scrum*

5.3.1. Origem

As raízes do *Scrum* estão em um artigo que sumariza as 10 melhores práticas em empresas japonesas, escrito por Takeuchi e Nonaka, cujo título é "*The New Product Development Game*", e foi publicado pela Havard Business *Review*, em fevereiro de 1986. Este artigo

introduziu o *Scrum* para referir-se às reuniões de equipes que praticam a auto-direção e a adaptabilidade.

No desenvolvimento de software, foi formalizado em 2005 por Ken Schwaber e Jeff Sutherland da Easel Corporation. O termo *Scrum* é o nome usado para a reunião de jogadores, no jogo de Rugby, quando eles se organizam em círculo para planejar a próxima jogada. É uma forma de mostrar que o projeto deve ser conduzido em pequenos ciclos, mas com uma visão de longo prazo, que é ganhar o jogo.

5.3.2. Pilares

O *Scrum* possui como base de sustentação, 03 pilares que devem trabalhar juntos para garantirem o bom e adequado funcionamento de todo o ciclo de vida do *Scrum*, como demonstra a Figura 5.1. São eles: transparência, inspeção e adaptação. Esses pilares são fundamentais e possuem como características dos valores e princípios do Manifesto Ágil.

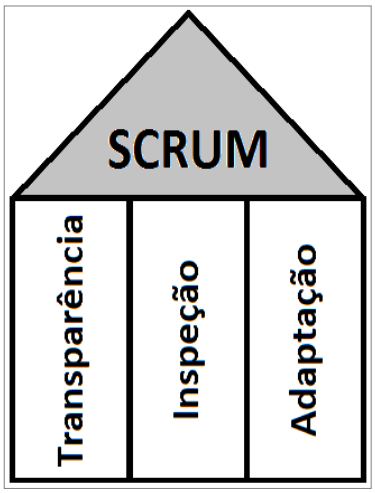


Figura 5.1. Os três pilares do Scrum

(Fonte: criação própria)

Os pilares do *Scrum* podem ser explicados da seguinte maneira:

- **Transparência:** todo e qualquer fator ou acontecimento relacionado ao processo de entrega, que possa impactar o resultado final do projeto (produto), deve ser visível e do conhecimento de todos envolvidos, inclusive o cliente. Este é um dos conceitos mais difícil de ser implementado, principalmente quando o *Scrum* está sendo adotado em projetos que utilizavam abordagens tradicionais de gestão de projetos.
- **Inspeção:** todos os aspectos do processo de entrega que possam impactar o resultado final do projeto devem ser inspecionados freqüentemente, para que qualquer variação prejudicial possa ser identificada e corrigida o mais rápido possível.
- **Adaptação:** toda vez que uma variação prejudicial é identificada, o processo deve ser ajustado imediatamente, como forma de evitar outros desvios.

Por ser um *framework*, o *Scrum* pode ser adaptado a qualquer tipo de projeto, seja ele de desenvolvimento de software ou não, desde que não comprometa os seus pilares.

5.3.3. Características do *Scrum*

O *Scrum* é um processo empírico bastante leve para gerenciar e controlar projetos de desenvolvimento de software e para criação de produtos, possuindo as seguintes características:

- Possui **equipes pequenas** de no máximo 09 pessoas;
- Utilizado preferencialmente em projetos que possuem os **requisitos instáveis** ou pouco conhecidos;
- **Iterativo e incremental** pelo fato de realizar pequenas entregas do projeto (incrementos) em pequenos ciclos e iterações chamadas de ***Sprint***, que possui duração de 02 a 04 semanas;
- As equipes são **Auto-gerenciáveis**, ou seja, os membros do Time devem ter autonomia na escolha de suas atividades, e devem decidir como e quando estas atividades serão implementadas, e qual o valor estimado para cada atividade;
- Por se tratar de um ***framework***, o *Scrum* ensina o que deve ser feito, e não em como fazer. Ou seja, ele possui as ferramentas e processos definidos, mas como ferramentas e processos serão utilizados vai depender das características de cada projeto.
- **Time-boxing** para todas as suas cerimônias e tudo o que houver necessidade de tempo. Ou seja, no *Scrum* sempre existe um tempo definido para realizar algo.

Por ser um *framework*, o *Scrum* pode ser adaptado a qualquer tipo de projeto, seja ele de desenvolvimento de software ou não, desde que não comprometa os seus pilares.

5.3.4. Os papéis

Uma das principais idéias do *Scrum* para eliminar a complexidade do desenvolvimento e gerenciamento de projetos é a implantação de um controle descentralizado, capaz de lidar mais eficientemente com contextos poucos previsíveis. Desse modo, não existe a figura única do gerente de projetos. Suas responsabilidades estão diluídas entre os seguintes papéis: *Product Owner, Scrum Master* e *Scrum Team* (Time).

- ***Product Owner:*** dono do produto ou o seu representante, com as responsabilidades de:
    - Definir **Visão do Produto**;
    - **Gerenciar** o retorno sobre o investimento (ROI);
    - **Apresentar** ao time os **requisitos necessários** para a entrega **do produto**;
    - **Priorizar cada requisito** de acordo com o seu valor para o negócio/cliente;
    - **Gerenciar a entrada de novos requisitos** e suas priorizações;
    - **Planejar** as entregas (releases);

- Atuar como **facilitador** quando mais de um cliente estiver envolvido no projeto;
- **Colaborar** com o Time.

- ***Scrum Master:*** responsável por direcionar o time para o sucesso, assegurando-se que o projeto e a cultura empresarial estão otimizadas para atender as expectativas do projeto. Assim, possui a responsabilidade de:
  - Garantir que o *Scrum* esteja sendo utilizado por todo o Time;
  - Ajudar a **manter o Time no Foco e Meta** definida para a *Sprint;*
  - Ensinar o cliente a **maximizar o ROI** e atingir seus objetivos através do *Scrum;*
  - **Priorizar e Remover** todo e qualquer **impedimento** do Time;
  - Facilitar as reuniões;
  - Combater o comando-controle;
- ***Scrum Team* (Time):** um Time *Scrum* deve:
  - Ser **auto-gerenciável**, de modo que possa se organizar e determinar a melhor estratégia de entrega com qualidade das funcionalidades de maior prioridade;
  - Ser **multi-disciplinar**;
  - Possuir entre 05 a 09 membros no Time;
  - **Motivado** e **comprometido** com o trabalho;
  - **Focado** em uma **Meta**;
  - **Comunicativo**;
  - Estar **organizado** em um **espaço físico adequado** para o trabalho;
  - Responsáveis pela **resolução de conflitos** e **pontos de melhoria**;

Uma maneira simples de explicar os papéis no *Scrum* é apresentada na figura abaixo:

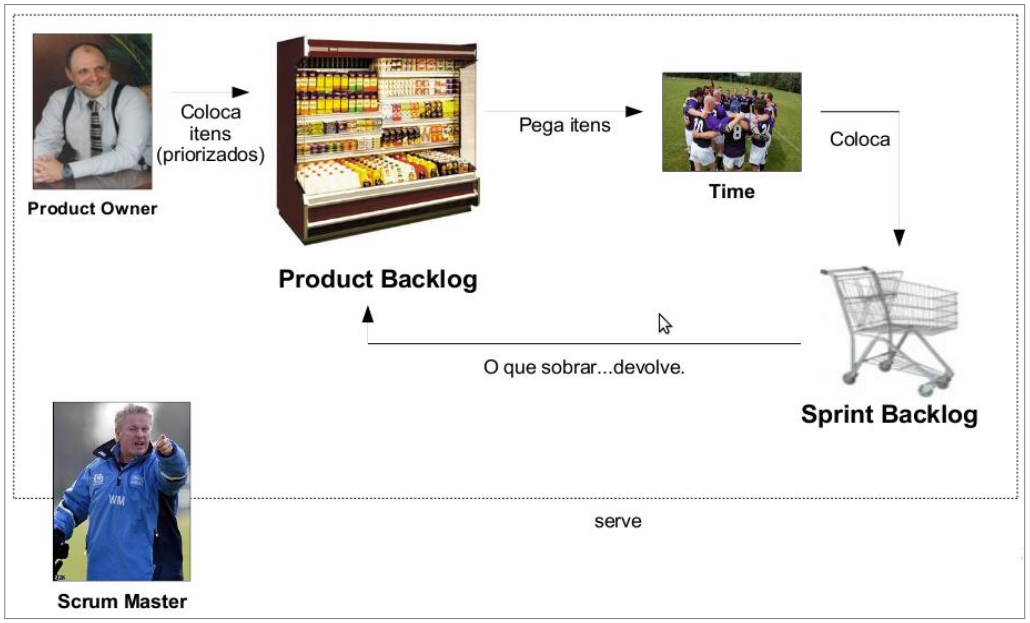


Figura 5.2. Fluxo simples explicando os papéis no Scrum

(Fonte: Magno, 2007)

Conforme se pode observar na figura acima, o *Product Owner* define os itens, e os coloca priorizados no Product *Backlog*, que contém todos os supostos requisitos que deverão ser desenvolvidos para o projeto. Esta lista de requisitos pode ser re-priorizada, e podem ser adicionados itens no Product *Backlog* a qualquer momento. O Time pega os itens que conseguem desenvolver na *Sprint* e coloca na *Sprint Backlog*. Os itens que são finalizados devem ser entregues ao *Product Owner*, e os não finalizados devem retornar ao Product *Backlog*. O *Scrum Master* deve acompanhar todo o processo, estando disposto o tempo todo para o Time para resolver qualquer tipo de impedimento, auxiliar também o *Product Owner* na definição e priorização dos itens a serem desenvolvidos, e garantir que todas as cerimônias estão sendo realizadas por todos do Time.

5.3.4.1. Os papéis x Atividades de Gerenciamento do Projeto

Como explicado anteriormente, no *Scrum* não existe uma figura única do gerente do projeto, este papel é diluído entre os três papéis do *Scrum.* Assim, cada papel possui a sua responsabilidade de gerenciamento.

O projeto no *Scrum* é dividido em *realeases* (podendo ser chamadas de marcos do projeto) que são formadas por um conjunto de iterações chamadas de *Sprints*. A linha de vida de um projeto *Scrum* pode ser representado como a figura abaixo.

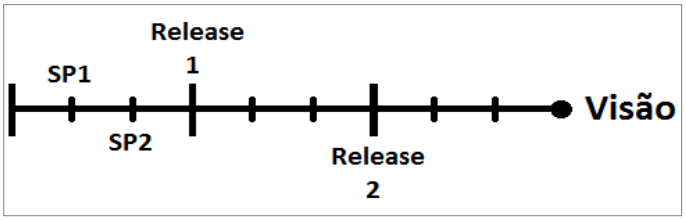


Figura 5.3. Linha de vida de um projeto Scrum

(Fonte: criação própria)

O gerenciamento do projeto a **nível Macro (*Macro Management*)** fica a cargo do ***Product Owner***, responsável por **gerenciar** e **planejar todo o projeto**, atribuindo uma visão do projeto para ser compartilhada com todos os comprometidos com o mesmo. Para gerenciar o projeto, o *Product Owner* deve acompanhar e verificar se as entregas das *Sprints* estão sendo realizadas de acordo com o planejado para o projeto, e tomar as providências cabíveis quando necessário.

O gerenciamento do projeto a **nível Micro (*Micro Management)*** fica sob a responsabilidade do Time. Este gerenciamento ocorre durante as *Sprints*, através de práticas utilizadas pelo *Scrum* como as reuniões de Planejamento (*Sprint Planning)*, as Reuniões Diárias (*Daily Meeting)*, a Revisão *(Sprint Review)* e a reunião de Retrospectiva (*Sprint Retrospective).* Portanto, cabe ao Time realizar o acompanhamento diário das *Sprints*, buscando sempre realizar as entregas do projeto, e encontrar soluções de melhorias para os desvios diários que possam vir a ocorrer.

O *Scrum Master* é o responsável por **gerenciar os processos** do *Scrum*, cabendo a este verificar se o processo adotado encontra-se adequado a realidade do Time, e se o mesmo está sendo seguido por todos do Time. Caso haja a necessidade, o *Scrum Master* tem a liberdade de realizar adaptações ao processo de modo não comprometer os três pilares do *Scrum.*

5.3.5. Artefatos

Durante o ciclo de vida de um projeto que adota o *Scrum* como *framework* de gerenciamento de projetos, alguns artefatos devem ser gerados. A seguir estão listados esses artefatos.

- ***Product Backlog:*** lista inicial das funcionalidades do produto definidas pelo *Product Owner*. Essa lista pode ser alterada (adicionar ou remover itens) durante todo o projeto. Nessa lista, o *Product Owner* definirá os itens que possuem maior prioridade para serem desenvolvidos.
- ***Sprint Backlog:*** lista de funcionalidades do *Product Backlog* selecionadas pelo Time para serem executadas durante a *Sprint*. As funcionalidades devem estar priorizadas, e podem estar dispostas em quadro de tarefas da *Sprint*, como apresentado na figura abaixo.

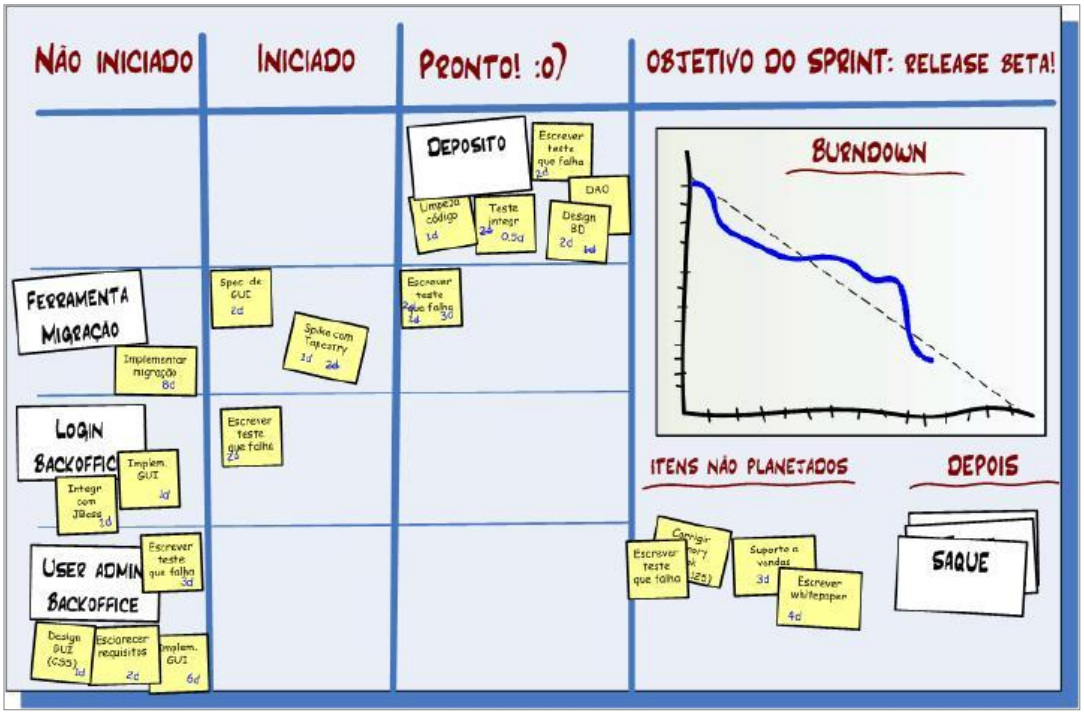


Figura 5.4. Exemplo de um Scrum Board para um Sprint

(Fonte: Kniberg, 2007)

- ***Burndown chart:*** gráfico que é utilizado em cada *Sprint*, atualizado diariamente, mostrando a produção do andamento do *Sprint*, em comparação a uma estimativa ideal de produtividade.

5.3.6. Cerimônias do *Scrum*

- ***Daily Meeting* (Reuniões diárias):** reuniões diárias entre os membros do Time e o *Scrum Master*. Devem ser realizadas em círculo de modo que todos se vejam, e deve ser em frente ao quadro do Time com as atividades que estão desenvolvendo. Esta reunião deve ser realizada em pé (*Stand Up Meeting)*, com duração de **15 minutos**, com o objetivo de sincronizar o trabalho diário do Time, encontrar impedimentos e pontos que merecem atenção durante a *Sprint*. A reunião deve ser muito objetiva, evitando conversas paralelas e detalhamento das atividades. Para isso, uma pessoa fala por vez, respondendo sempre as 03 perguntas:
  - *O que você fez desde a última reunião?*
  - *O que você vai fazer hoje?*
  - *Existem impedimentos?*

  Caso exista um impedimento, o *Scrum Master* deve anotá-lo, e conversar somente com os envolvidos neste para solucioná-lo. Existem casos em que é necessário o agendamento de uma reunião para discutir sobre o assunto. Na figura a seguir encontra-se um exemplo de uma reunião diária de um Time *Scrum*.

Figura 5.5. Exemplo de uma reunião diária

- ***Sprint Planning (Planejamento da Sprint):*** reunião de planejamento, que tem a participação de todo o time para decidir o que vai ser trabalhado na *Sprint* (*Selected Product Backlog*). A escolha dos itens que vão ser trabalhados na *Sprint* deve ser decidida com base na estimativa para cada item do *Product Backlog*. A estimativa de cada item pode ser realizada colaborativamente utilizando a técnica de *Planning Poker*.
- ***Sprint Review* (Revisão da *Sprint*):** reunião que ocorre ao final de cada *Sprint*, com todo o time, com duração de 04 horas para *Sprints* de um mês, onde será apresentado o resultado do trabalho da *Sprint*, para o *Product Owner* e demais interessados, a fim de obter um feedback e determinar quais são os objetivos da próxima *Sprint*.
- ***Sprint Retrospective* (Retrospectiva da *Sprint*):** reunião que ocorre com o Time após o *Sprint Review*, cujo objetivo é revisar o processo de trabalho. Esta reunião tem duração de 03 horas para *Sprints* de um mês. Deste modo, cada membro do Time deve responder a algumas perguntas, que serão discutidas para melhorar o ciclo do próximo *Sprint*.
    - *O que fizemos bem durante a Sprint que devemos continuar fazendo?*
    - *O que pode ser melhorado para a próxima Sprint?*
    - *Quais são as ações a serem executadas para que possamos melhorar o que não foi bom?*

5.3.7. Fluxo do *Scrum*

O ciclo de vida do *Scrum* conta com a participação de todos os membros do projeto, diluídos juntamente com os seus papéis (*Product Owner, Scrum Master, Scrum Team).* E nele devem ser gerados todos os artefatos do *Scrum*, assim como devem ser realizadas todas as práticas propostas pelo *Scrum*, de modo que os pilares do *Scrum* sejam cumpridos.

A imagem abaixo representa o ciclo de vida de um projeto *Scrum* que é explicado a seguir.

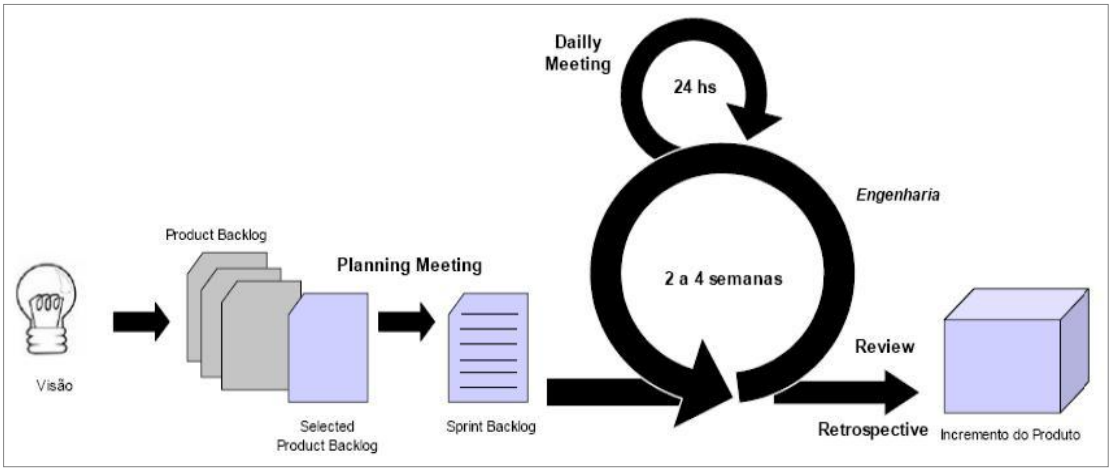


Figura 5.6. Ciclo de vida de um projeto Scrum

(Fonte: adaptado de Kniberg, 2007)

O projeto com o *Scrum* deve iniciar primeiramente com a divisão dos papéis entre os integrantes do projeto. Assim, deve ser escolhido o *Product Owner, Scrum Master* e os membros do Time.

5.3.7.1. Definindo a Visão do Produto

Na **primeira reunião** do projeto o *Product Owner* deve definir e explicar para todos os envolvidos no projeto, a **Visão do Produto** a ser desenvolvido. Esta Visão é o que representa a sua necessidade que deverá ser entregue ao fim do projeto. Para definir a Visão o *Product Owner* colhe informações junto a clientes, usuários, time, gerentes, *stakeholders*, executivos, etc.

É importante que os itens da Visão do Produto sejam uma regra que deve ser seguida ao longo do projeto. Logo, esta Visão deve estar bem definida pelo *Product Owner*, e deve ser muito bem entendida pelo Time. Caso exista necessidade, esta visão poderá ser refinada durante esta reunião.

Após a definição da Visão do Produto, deve-se escolher o **tamanho da *Sprint***. O *Scrum* define que uma *Sprint* deve possuir duração de 02 a 04 semanas. Portanto, o tamanho ideal de uma *Sprint* é o tamanho que seu Time e o *Product Owner* acham ideal.

Outro ponto bastante importante que deve ser definido em um projeto é o conceito de **Pronto**. Pois o *Scrum* exige que os Times construam um incremento de funcionalidade do produto potencialmente pronto para o *release* ao final de cada *Sprint* Portanto um item

entregue deve estar “pronto” para ser utilizado com outros incrementos entregues, que devem ser passíveis de serem testados, garantindo que todos os incrementos funcionam juntos adequadamente. Sendo assim, deve estar bastante claro no inicio do projeto quando um item será considerado pronto.

5.3.7.2. Definindo o *Product Backlog*

O *Product Owner* deve definir a lista de funcionalidades ou requisitos que devem ser realizadas durante o projeto. Um uso muito comum no *Scrum* é definir esta lista de requisitos como **Histórias de Usuário** (*User Story)* que são pequenas descrições de funcionalidades fornecidas pelo cliente ou *Product Owner,* objetivando gerar uma visão compartilhada de negócio e técnica para todos os envolvidos no projeto.

A lista de funcionalidades deve estar priorizada e deve ser mantida e gerenciada pelo *Product Owner.* Ou seja, é dever do *Product Owner* inserir os itens que devem ser desenvolvidos no *Product Backlog*. Existem casos em que poderão surgir itens técnicos do Time que deverão entrar na lista de requisitos do *Backlog,* no entanto, o *Product Owner* deve estar ciente e de acordo com a inclusão destes itens.

É importante salientar que um *Product Backlog* nunca está completo, ele evolui à medida que produto em si e o ambiente em que o *Backlog* está sendo utilizado se desenvolve. Ele é dinâmico no sentido de que ele está constantemente mudando para identificar o que o produto precisa para ser apropriado, competitivo e útil.

Quando afirmamos que as funcionalidades devem estar priorizadas, queremos dizer que pelo menos os itens que o *Product Owner* deseja, ou acredite que existe uma remota possibilidade destes itens serem desenvolvidos na próxima *Sprint*, estes devem estar priorizados com uma escala única de importância.

5.3.7.3. Planejamento da *Sprint*

O planejamento de uma *Sprint (Sprint Planning)* é um dos principais eventos de um projeto com o *Scrum.* Neste momento a iteração do projeto é planejada, dando-se início a iteração (*Sprint)* em si. Esta reunião deve possuir um local, horário, participantes, e tempo de duração previamente definidos.

O *Scrum* define que para *Sprints* de 01 mês esta reunião deve ter no máximo 08 horas de duração. E para as *Sprints* mais curtas, o tempo de duração deve ser proporcional. É importante salientar que o *Scrum* trabalha com tempo definido (*Time Boxing)* que deve ser respeitado por todos.

Para o planejamento o Time, *Scrum Master* e o *Product Owner* devem estar presentes na reunião. Esta reunião deve iniciar com o *Product Owner* definindo a meta para o projeto que deve ser alcançada na *Sprint.* Deste modo, deve explicar quais os requisitos devem ser entregues para que esta meta seja atingida.

A meta da *Sprint* deve ser uma descrição que fornece orientação ao Time sobre a razão pela qual ele está produzindo o incremento. O motivo para se ter uma Meta da *Sprint* é permitir ao Time um espaço para variação em se tratando de funcionalidade. É

importante saber que esta meta deve ser sempre relacionada ao negócio e não devem ser numéricas ou representar a desempenho a ser alcançado pelo Time.

A reunião de planejamento é dividida em duas partes (como demonstra a figura abaixo): a parte 1 deve tratar do planejamento estratégico, e na segunda parte o planejamento tático.

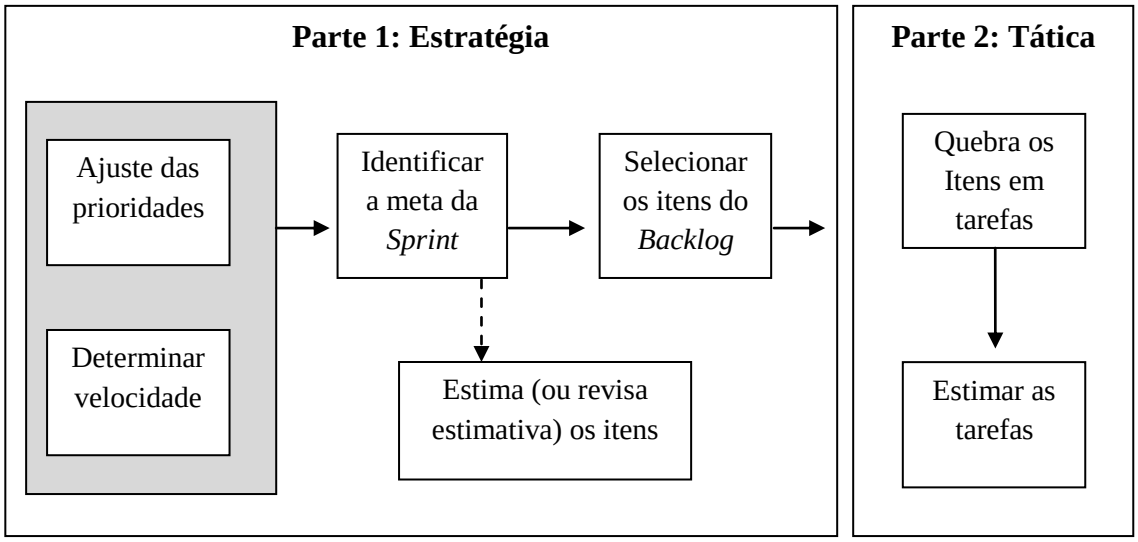


Figura 5.7. Divisão da *Sprint Planning*

- **Parte 1: Planejamento Estratégico**

Nesta reunião devem estar presentes o *Product Owner*, os membros do Time e o *Scrum Master*. A reunião deve ser iniciada pelo *Product Owner* informando o seu objetivo para a *Sprint* e as histórias mais importantes. Nesta reunião podem se realizados os ajustes das prioridades do *Backlog*. No entanto, seria importante que antes de iniciar esta reunião, o *Product Owner* olhasse o *Product Backlog* para verificar se as suas prioridades estão adequadas, realizando os ajustes quando necessários.

Após, o Time deve apresentar ao *Product Owner* a velocidade (***Velocity)*** com a qual conseguem trabalhar (tamanho da quantidade máxima de pontos que conseguem entregar por *Sprint*, que deve ser calculada pela média histórica da soma de Pontos Entregues em cada *Sprint),* e em seguida realiza/revisa as estimativas sobre as histórias baseadas no esforço estimado para cada história mais importante (***Selected Product Backlog)*** do *Product Backlog* para o *Product Owner* para aquela *Sprint.*

Existem diversas técnicas de estimativas que podem ser utilizadas em projetos *Scrum*. O Planning Poker é uma das mais populares, onde, utiliza-se cartas com uma numeração seguindo uma sequência parecida com a tabela de Fibonacci. Após a estimativa dos itens do *Backlog*, e com base na velocidade de entrega do Time, as histórias são selecionadas pelo Time para comporem a *Sprint*, chamada de ***Sprint Backlog***. É importante lembrar que a decisão do que vai ser selecionado compete ao Time, pois somente ele é capaz de informar a sua capacidade de entrega para a *Sprint.*

É importante que o Time pergunte ao *Product Owner* se com os itens que foram selecionados para *Sprint* a meta vai ser cumprida.

- **Parte 2: Planejamento Tático**

Reunião de caráter mais técnico, devendo participar os membros do Time, o *Product Owner,* e o *Scrum Master*. A presença do *Product Owner* na reunião é apenas para esclarecer eventuais dúvidas sobre os itens, não devendo opinar na discussão técnica. Ainda, poderão ser convocadas outras pessoas para a reunião, de acordo com a necessidade do Time.

Nesta reunião deverão tratar a questão do "como?", ou seja, o Time deve discutir e decidir como fazer para transformar o *Selected Product Backlog* em um incremento pronto e desenvolvido ao final da *Sprint*. Para isso deve-se projetar o que deverá ser feito, identificando pequenas tarefas a serem desenvolvidas para cada história em discussão.

Para elaborar as tarefas, as histórias devem ser discutidas uma a uma, por todos os membros do Time. É importante que as tarefas sejam decompostas para que possam ser realizadas em menos de um dia, de modo a permitir uma visualização mais real do andamento da *Sprint.* As tarefas devem ainda ser estimadas, onde o Time poderá escolher como isso irá ocorrer: estimativa em horas, ou distribuição dos pontos da história para as tarefas.

5.3.7.4. Execução da *Sprint*

Quando finalizada a reunião de Planejamento da *Sprint*, deve-se iniciar a execução da *Sprint*, que tem duração de 02 a 04 semanas. Esta etapa é dividida em:

1. **Montagem do quadro da *Sprint*:** a primeira parte a ser iniciada é a montagem do quadro do Time a ser utilizado para a *Sprint.* Um exemplo (o quadro pode ser montado seguindo as particularidades de cada Time) de quadro é apresentado na figura a seguir, que é formado por 04 colunas, explicadas a seguir:
    - **a)** A coluna ***Sprint Backlog*** representa os itens selecionados do *Product Backlog* para serem trabalhadas durante a *Sprint*;
    - **b)** A coluna ***To Do*** são as atividades que deverão ser desenvolvidas para cada item.
    - **c)** ***Doing*** representa a coluna em que são colocadas as tarefas que estão sendo trabalhadas no momento.
    - **d)** ***Done*** representa a coluna das tarefas que foram finalizadas. Quando todas as tarefas de uma história são finalizadas, a história também deverá passar para a coluna ***Done***.

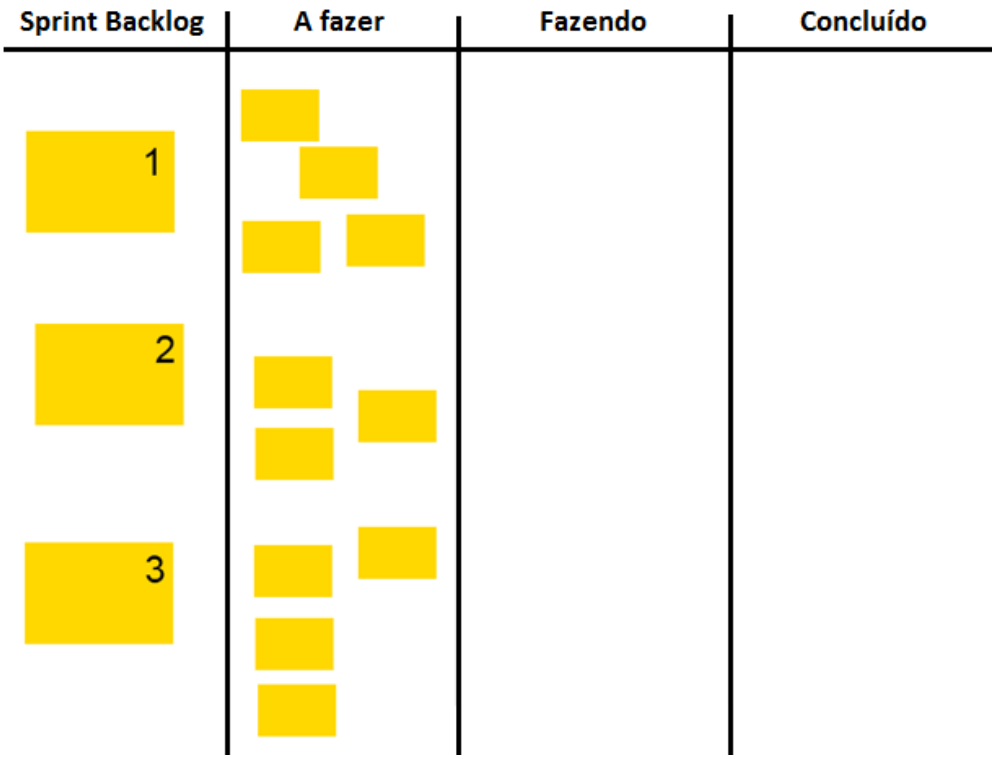


Figura 5.8. Exemplo de um quadro Scrum para uma Sprint

(Fonte: criação própria)

2. **Definição da Data e Hora das reuniões:** ao iniciar uma *Sprint* deve-se definir o horário e local das Reuniões Diárias (*Daily Meeting)*, da Revisão da *Sprint (Sprint Review)* e a da reunião Retrospectiva da *Sprint (Sprint Retrospective).* Esse é um dos pontos importantes para garantir que todos os interessados participem e não esqueçam os horários das reuniões.

3. **Desenvolvimento da *Sprint:*** as histórias selecionadas deverão ser desenvolvidas até o final da *Sprint* pelo Time. Para isso o Time deverá possuir ao seu dispor um ambiente adequado para permitir o desenvolvimento do projeto. A maneira como o Time se organiza para desenvolver as histórias e tarefas ficam a critério do Time. As histórias deverão ser desenvolvidas de acordo a sua prioridade, que geralmente dispomos as histórias mais prioritárias no topo da *Sprint Backlog*.

4. **Acompanhamento:** durante a execução da *Sprint* acompanhamentos serão realizados nas reuniões diárias por todo o Time, e caso exista alguma história com algum impedimento que esteja impossibilitando a sua execução/finalização, o membro do Time deverá informar, e o *Scrum Master* deverá procurar uma solução, ou caso seja um impedimento técnico, o Time que deverá ajudar na resolução do impedimento. Diariamente o time deverá atualizar o andamento de suas tarefas no quadro, e também atualizar o gráfico de *BurnDown* após as reuniões diárias. É importante que o *Scrum Master* esteja disponível e atento aos acontecimentos, não apenas durante as reuniões, mas durante toda a *Sprint*, de modo a perceber e verificar se o andamento da *Sprint* está conforme o planejado, ou se novos direcionamentos serão necessários.

5. **Finalização:** ao final da *Sprint* o Time deverá apresentar ao *Product Owner* o resultado de tudo o que foi realizado durante a *Sprint*.

5.3.7.5. Revisão da *Sprint* (*Sprint Review*)

O último dia da *Sprint* deve ter a entrega do incremento para o produto, com a apresentação dos itens selecionados para a *Sprint* e que foram finalizadas durante a execução da *Sprint.*

Esta reunião deve ser realizada no dia e hora agendada no início da *Sprint*, e deverá contar com a participação de todos os membros do Time, do *Scrum Master*, do *Product Owner*, e qualquer pessoa interessada, desde que sua participação for aceita pelo *Product Owner.*

O time deverá ler a descrição item a item que foi selecionado para a *Sprint*, e um a um deverá apresentar o que foi feito. Durante a apresentação, questionamentos e testes poderão ser realizados sobre o que foi implementado. É importante que sejam anotadas as considerações sobre cada história implementada.

Ao final da apresentação de cada história o *Product Owner* deve dar o seu aval a respeito do que foi desenvolvimento, informando se atende ou não ao que foi solicitado. Assim, as histórias podem ser: aceitas, aceitas com restrições, ou rejeitadas.

Quando uma história for "aceita", o total de pontos estimado para ela deve ser contabilizado para a *Sprint.* As histórias poderão ser consideradas "aceitas com restrições" somente se a sua correção for rápida, não passando mais do que uma hora. O total de pontos entregues pelo Time na *Sprint* deverá ser a soma de todas as histórias aceitas e aceitas com restrições. As histórias rejeitadas não são contabilizadas os pontos, e devem ser anotados os motivos para esta situação, para que possam ser corrigidas em outro momento.

As histórias que não foram concluídas deverão não precisam ser apresentadas ao *Product Owner.* No entanto, ao final o *Product Owner* colocar estas histórias e repriorizá-las, juntamente com as rejeitadas novamente no *Product Backlog* se a implementação destas ainda forem de sua necessidade.

5.3.7.6. Retrospectiva da *Sprint* (*Sprint Retrospective*)

Ao finalizar a *Sprint Review*, o Time e o *Scrum Master* deverão se reunir por três horas (para *Sprints* de um mês) para realizarem uma retrospectiva da *Sprint*. Essa reunião representa o espírito de Inspeção e Adaptação dentro do *Scrum*. A reunião geralmente é conduzida pelo *Scrum Master* que deve encorajar todos os membros do Time a revisar todo o processo de trabalho, o que envolve: o produto do trabalho, como foi desenvolvido, as práticas do *Scrum*, e a relação entre as pessoas.

A inspeção deve identificar três pontos importantes que devem ser discutidos entre os membros do Time e o *Scrum Master*:

- **O que fizemos bem:** o que foi que deu certo para esta *Sprint* que se pudéssemos faríamos da mesma maneira?

- **O que precisamos melhorar:** se tivéssemos que fazer outra *Sprint,* o que faríamos de uma maneira diferente?

- **Ações:** quais são as possíveis ações de melhorias que podemos implementar no futuro? As ações a serem tomadas devem ser priorizadas, e devem possuir um responsável para cada uma delas.

Figura 5.9. Exemplo de uma *Sprint Retrospective*

5.4. Dicas para executar o *Scrum* na prática

Com base na experiência do uso do *Scrum* em projetos por parte dos autores deste capítulo, abaixo estão descritas um conjunto de dicas que podem ser úteis durante a execução de um projeto que utiliza o *Scrum*.

5.4.1. Como definir a Visão do Produto?

Definir uma Visão de um Produto pode não ser uma tarefa fácil de ser elaborada. Portanto para que esta seja uma Visão efetiva, procure responder às seguintes perguntas:

- ✓ Quem irá comprar este produto? Quem é o cliente alvo? Quem irá usar o produto? Quem são os usuários alvo?
- ✓ Quais são necessidades do cliente (ou usuários) que o produto pretender resolver? Qual o valor o produto adicionará?
- ✓ Quais são os atributos que o produto precisa possuir para suprimir estas necessidades, e quais garantirão o sucesso do produto?
- ✓ Como o produto pode ser comparado a produtos ou alternativas existentes? Quais são os pontos diferenciais deste produto?

Existem situações em que o *Product Owner* não consegue definer a Visão do Produto, assim o *Scrum Master* deve auxiliá-lo na criação da mesma.

5.4.2. Como criar uma História de Usuário?

Uma História de Usuário, do inglês *User Story*, é uma técnica para se representar requisitos em um *Product Backlog*. Ela descreve uma funcionalidade que deve fornecer valor para usuários ou cliente de um projeto.

As histórias de usuários são compostas de 03 aspectos:

- ✓ Uma descrição da história para ser usada nas sessões de Planejamento, e também como uma lembrança do desejo do cliente.
- ✓ Conversas sobre a história que servirão como um "flash" sobre a mesma.
- ✓ Testes que documentam os detalhes da história e podem ser usados para determinar quando ela está completa.

As histórias de usuário não precisam ser muito detalhadas por escrito, pois o desenvolvimento ágil tem a premissa da comunicação constante entre o Time o *Product Owner*. Ou seja, as informações detalhadas sobre as histórias devem ser obtidas em conversas formais durante as reuniões de planejamento, ou informalmente quando houver a necessidade.

Para criar uma História de Usuário podemos seguir os passos da imagem abaixo.

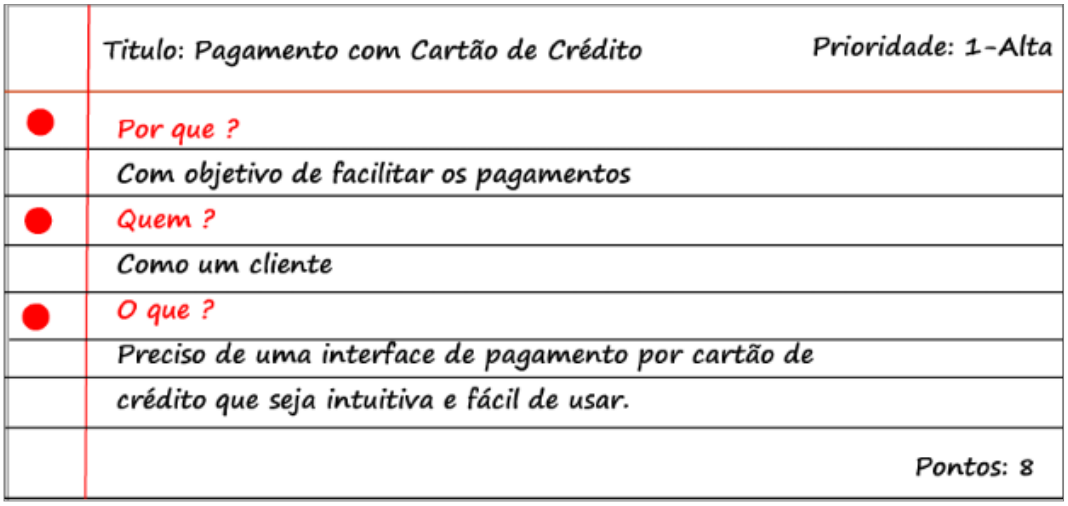


Figura 5.10. Exemplo de como criar uma História de Usuário

Primeiro deve-se ter um **título para a História**, de modo que este seja único, e descreva o contexto da funcionalidade a ser desenvolvida. Depois responda as seguintes perguntas para elaborar a **descrição para a história**:

- ✓ Por que eu quero esta funcionalidade?
- ✓ Quem é que deseja esta funcionalidade?
- ✓ O que eu preciso para esta funcionalidade?

Depois para cada história, deve-se atribuir um **grau de prioridade** para mesma. Uma boa técnica é utilizar esta prioridade em forma de números, atribuídos de 1 à 100, de maneira que cada história possua um valor de negócio diferente.

Além da descrição de uma história de usuário, é importante que exista a definição dos **critérios de aceitação** de cada história pelo *Product Owner*, de modo orientar o desenvolvedor na execução das atividades desenvolvidas para uma história. Estes critérios de aceitação devem ser apresentados e informados durante a apresentação da *Review* ao final da *Sprint.*

5.4.3. Como definir o tamanho de uma Sprint?

A seguir serão apresentados alguns cenários, e a escolha do tamanho da *Sprint* de acordo com estes cenários.

- Mudança constante no topo do *Product Backlog:* ideal trabalhar com *Sprints* curtas.
- Síndrome do estudante presente na equipe: ideal trabalhar com *Sprints* curtas.
- Dificuldade de entregar valor para o cliente ao fim das *Sprints:* ideal trabalhar com *Sprints* curtas.
- Time e/ou clientes exaustos com loops tão curtos: ideal trabalhar com *Sprints* longas.

Portanto, é importante que a definição do tamanho seja realizada em acordo comum entre o Time e o *Product Owner*, podendo este tamanho sofrer alterações ao longo do tempo. Contudo, é importante que este tamanho seja testado na prática por mais de uma *Sprint*, ao invés do tamanho ser alterado com freqüência, pois se for alterado com freqüência, o Time não consegue obter um valor mais preciso de referência para o tamanho da *Sprint.*

5.4.3.1. Definindo o conceito de “Pronto” para uma história

A definição e entendimento comum do conceito de “pronto” (*Done*) de ser partilhado por todos do projeto. Tanto para os que executam o trabalho, o Time, quanto para os que validam os seus resultados, o *Product Owner.* Isso é fundamental o funcionamento adequado do *Scrum*

A definição de “Pronto” no desenvolvimento de software podem ser uma lista simples de tarefas que devem ser realizadas para uma determinada história: escrever o código, criar os testes unitários, criar os testes de integração, executar o incremento no ambiente de produção, etc.

Definir quais são os passos que devem ser realizados para que uma história esteja “Pronta” é permitir que a história seja auditável, e seja possível verificar se todas as atividades necessárias de uma história foram concluídas de modo a agregar valor ao produto.

5.4.3.2. Como estimar histórias usando a técnica de *Planning Poker*?

A técnica de *Planning Poker* visa realizar a estimativa dos itens do *Backlog* de maneira colaborativa entre todos os membros do Time. Para isso, cada membro possui um baralho de 13 cartas (veja a imagem a seguir), cujos valores são: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, ?, Stop. Estes valores são chamados de Pontos por História (*Story Points).*

Figura 5.11. Exemplo de cartas utilizadas para o *Planning Poker*

Para realizar a estimativa de uma história de usuário (*User Story)*, cada membro do Time escolhe um valor de carta que acredita representar o tamanho para aquela história. Para isso, os membros do Time precisam de algum tipo de entendimento sobre o quê se trata a história. E como todos os membros devem estimar cada item, certificamos que cada membro do Time possui entendimento para desenvolver o item.

As primeiras cartas que devem ser entendidas são: 0 (zero) representa um item que não tem quase nada a ser feito, está praticamente concluído; a interrogação (?) indica que o membro do Time não possui informação suficiente para estimar o item; e o símbolo do café representa que devemos parar a estimativa, e descansar (essa é uma carta incluída depois, não é original da prática do *Planning Poker).*

Portanto, como iniciar o processo quando ainda nunca realizamos uma estimativa com esta técnica? Primeiro identifique o item da *Sprint Backlog* que todos do Time concordam em ser o item de menor esforço, o mais simples de ser resolvido. Ao chegar a um consenso, atribua o valor 2 a este item. Por que o valor 2? Porque podem existir situações em que durante o andamento das estimativas, o Time identifique que existia um item ainda mais simples de ser implementado, e a este atribuiremos o valor de 1.

Após a definição do item mais simples, escolhe-se o primeiro item da *Sprint Backlog*, e realiza uma comparação de esforço entre esta história, e a outra em que foi atribuída o valor de 2. **Pergunta-se:** quantas vezes mais esta história é maior que a anterior? Ou seja, sempre teremos um referencial de comparação entre as histórias. Assim, deve-se escolher uma única carta do baralho, e nunca devem ser somados os valores das cartas. Cada membro deverá colocar a sua carta virada para baixo, e após todos escolherem um valor é que os valores das cartas poderão ser mostrados.

Ocorrendo de existir valores diferentes, o quê deve-se fazer? Apenas os membros que atribuíram o **menor valor de esforço**, e os que atribuíram o **maior valor de esforço** devem explicar os motivos de suas estimativas. Após as explicações, deve-se iniciar uma nova rodada de estimativa envolvendo todos os membros novamente, e o processo se repete novamente, até que os valores estimados ao item cheguem a um consenso. Pode haver situações em que é difícil encontrar um consenso, devendo assim ser escolhido o valor atribuído pela maioria.

Portanto, o *Planning Poker* estimula o dialogo durante as rodadas de estimativa, pois os membros explicam aos outros o porquê estimaram o item com aquele tamanho. Desse modo, todo este processo gera um compartilhamento de conhecimento entre os membros do Time.

5.4.4. Como montar um gráfico de Burndown?

Um dos artefatos gerados pelo uso do *Scrum* é o gráfico de Burndown, que podem ser classificados em:

- ***Sprint Burndown Chart**:* este gráfico representa o número de pontos restantes (a ser realizado) com relação tempo restante de desenvolvimento para a *Sprint.* O **trabalho restante (pontos)** a ser realizado é representado pela **coordenada Y,** e o **tempo restante** de desenvolvimento é representado pela **coordenada X**. A figura a seguir apresenta esse tipo de gráfico, onde a linha na **cor laranja representa os pontos que estão sendo entregues a cada dia**, e a **linha na cor vermelha representa a linha ideal de entrega dos pontos**.

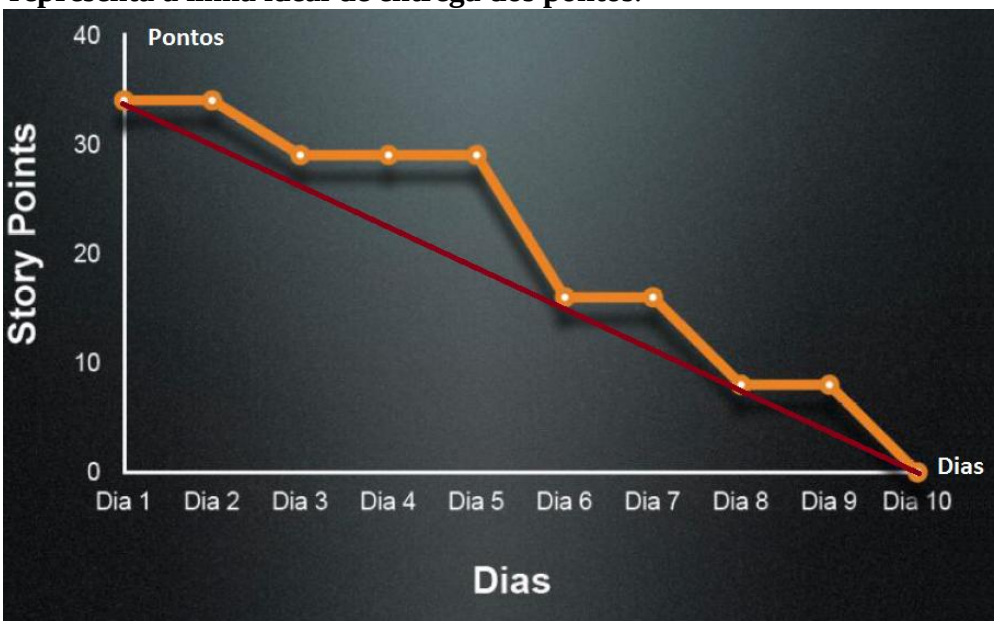


Figura 5.12. Exemplo de um Burndown Chart da Sprint

- ***Release Burndown Chart**:* este gráfico representa o número de pontos restantes a ser entregue na *release* de um projeto. Este tipo de gráfico só é possível de ser realizado quando o *Product Backlog* está completamente definido, e todas as histórias foram estimadas. O **trabalho restante (pontos)** a ser realizado é representado pela **coordenada Y,** e o total de ***Sprints* restantes** do release é

representado pela **coordenada X**. A figura a seguir apresenta esse tipo de gráfico, onde a linha na **cor laranja representa os pontos que estão sendo entregues a cada *Sprint***.

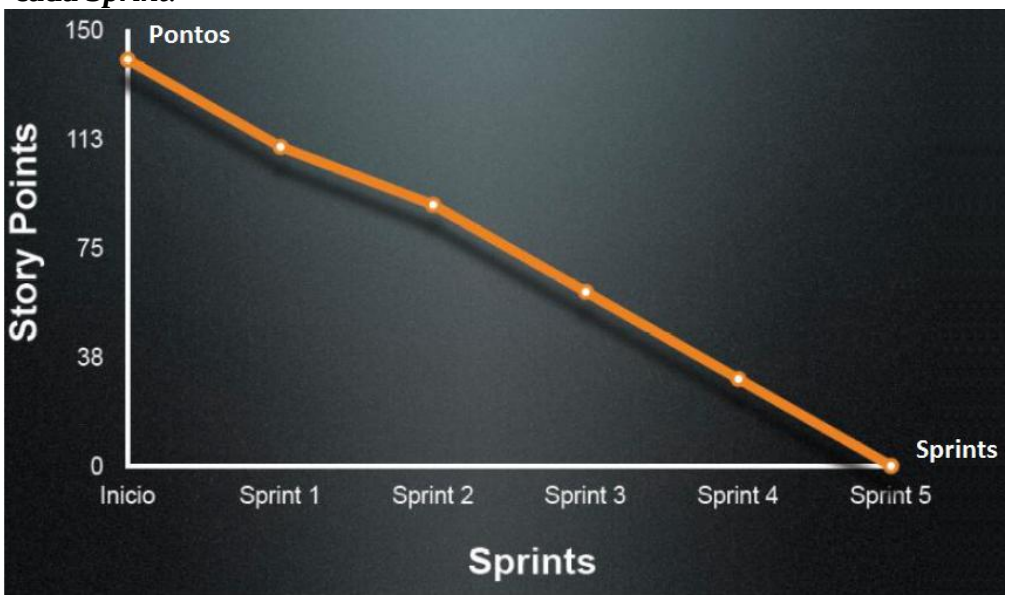


Figura 5.13. Exemplo um gráfico de Release Burndown

- ***Velocity Chart**:* este gráfico representa o número de pontos restantes entregue em cada *Sprint* realizada. Onde os **pontos entregues** é representado pela coordenada Y, e as **Sprints realizadas** é representada pela coordenada X.

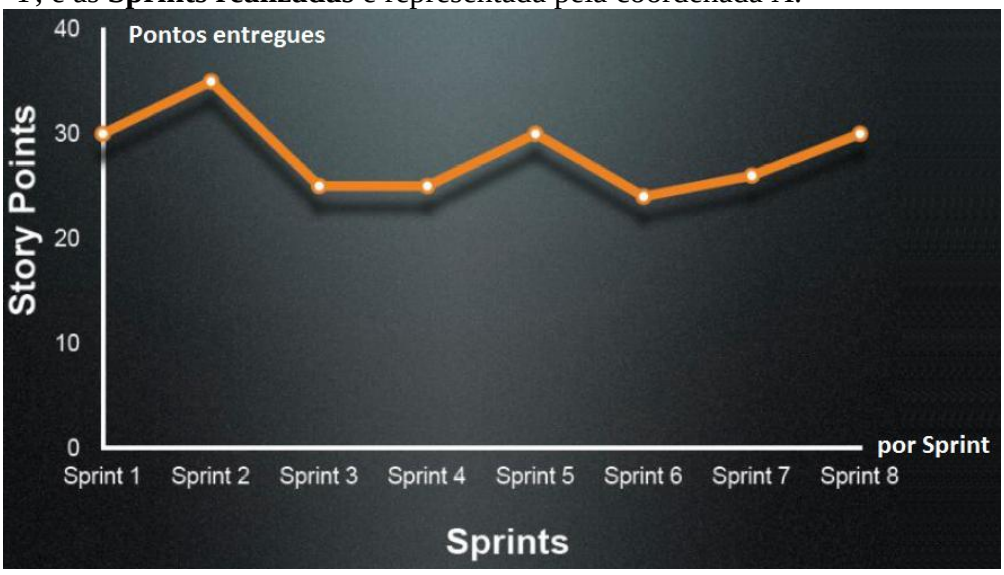


Figura 5.14. Exemplo de um gráfico de Velocity

- ***Tasks Burndown Chart**:* este gráfico representa o número de horas restantes (estimada) para as tarefas da *Sprint* com relação tempo restante (dias) de desenvolvimento para a *Sprint.* A **quantidade de horas** a ser realizado é representado pela **coordenada Y,** e o **tempo restante (dias)** de desenvolvimento é representado pela **coordenada X**. A figura a seguir apresenta esse tipo de gráfico, onde a linha na **cor laranja representa as horas que estão restando das tarefas,** realizadas a cada dia.

Figura 5.15. Exemplo de Tasks Burndown Chart

5.4.4.1. Como montar um ambiente adequado para um Time Scrum?

Para desenvolver o trabalho dentro do tempo definido em uma *Sprint*, é importante que o Time tenha ao seu dispor, um ambiente adequado para o Time. Com a participação em muitos projetos que utilizam o *Scrum* percebemos que:

- O Time deve possuir um ambiente próprio para discussão sobre o design do projeto, onde possam escrever, apagar, e deixar as anotações ou lembretes visíveis. Sendo assim, é importante que no ambiente esteja disponível um grande quadro branco, além do quadro de tarefas do Time.
- O ambiente deve ser descontraído para o Time, onde ele se sinta a vontade.
- O time deve estar sempre junto, pois isso permite uma maior interação e envolvimento entre todos os membros do Time. Estarem juntos significa:
    - Audibilidade: qualquer membro do Time pode conversar com outro sem precisar gritar ou sair de sua mesa.
    - Visibilidade: todos os membros do Time conseguem ver os demais, e ainda conseguem enxergar o quadro de tarefas da *Sprint* com facilidade.
    - Isolamento: se houver uma discussão sobre implementação de código ou algo referente ao projeto entre os membros do Time, nenhuma outra equipe será perturbada pelo barulho da discussão. E vice-versa.
- O time deve ter ao seu dispor um ambiente livre de interferências externas. Toda e qualquer interferência deve ser filtrada pelo *Scrum Master* que tem o papel de proteger o Time de interferências de modo garantir que possam focar 100% na meta da *Sprint.*

5.4.4.2. Como garantir uma reunião de Revisão (*Sprint Review)* adequada?

As reuniões de Revisão de uma *Sprint (Sprint Review)* são importantes para um projeto, pois é nesta reunião que o Time apresenta o resultado de tudo o que foi desenvolvimento

em certo período de tempo para o projeto. Nesta reunião deverão estar presentes todos os interessados no projeto: membros do Time, *Scrum Master, Product Owner*, e qualquer interessado ou convocado para esta reunião.

Como uma reunião importante, deve-se estar atento aos seguintes pontos:

- As reuniões de Revisão (*Review)* devem possuir data, horário e local definido logo no início da *Sprint.*
- No horário da reunião o Time deve estar pronto para apresentar o que foi feito, ou seja, tudo já deve estar integrado e preparado.
- Todos os presentes na reunião devem saber qual o objetivo da mesma. Caso contrário, dedique um tempo para explicá-la.
- O *Product Owner* deve estar ciente de todas as histórias que foram aceitas pelo Time para a *Sprint.*
- O Time deve apresentar algo realmente funcional, não havendo necessidade de apresentar algo que não foi concluído, ao menos que seja solicitado pelo *Product Owner*.
- O *Product Owner* deve dar um *feedback* sobre todas as histórias apresentadas.
- O Time deve estar pronto para demonstrar e permitir que as pessoas testem o que foi desenvolvido.

5.4.4.3. Como garantir uma reunião de Retrospectiva (*Sprint Retrospective)* adequada?

As reuniões de Retrospectiva atendem os 03 pilares do *Scrum*, e para que estas se tornem realmente efetivas alguns pontos importantes são listados a seguir:

- O ambiente da reunião deve ser confortável, geralmente fora do ambiente de trabalho, e sem qualquer tipo de interrupção.
- Uma pessoa deve ser responsável por ser anotar os pontos discutidos. Geralmente utilizamos um quadro branco para permitir um aprofundamento nas discussões.
- Ao iniciar uma reunião de retrospectiva é importante que se verifique se todos os pontos a serem melhorados na *Sprint* anterior foram realizados.
- Os principais pontos positivos e negativos devem ser enumerados. Podem ainda ser realizadas votações para identificá-los.
- Todos os pontos que devem ser melhorados devem possuir ações específicas, devem possuir um responsável para executá-la e a sua prioridade de execução.
- Deve-se verificar se todos os pontos de melhoria são possíveis de serem resolvidos até a próxima reunião de Retrospectiva.
- Todos os membros devem se sentir bastante confortável para dialogar durante a reunião. Existem casos em que é preciso o estímulo por parte do *Scrum Master* para que isso ocorra.
- O *Scrum Master* deve evitar que apenas algumas pessoas tomem conta da reunião.

- É importante que tragam informações em números sobre a *Sprint* para serem discutidos, como: pontos entregues, custo do ponto, etc. E ainda realizem um comparativo com as *Sprints* anteriores, objetivando encontrar algum ponto fora da curva, seja ele positivo ou negativo.
- Ao final da reunião, deve-se realizar um breve resumo do que foi discutido, e identificar os principais pontos de atenção.

5.5. Indicadores de Qualidade

Além dos gráficos apresentados, outros indicadores de qualidade podem ser obtidos durante o projeto, de forma auxiliar na inspeção em busca de possíveis melhorias. Alguns destes indicadores podem ser:

- Efetividade do Time: indicador que informa o quanto um Time está sendo efetivo para implementar o que foi planejado para a Sprint.

$$\frac{\textbf{Número de pontos planejados } \textit{na Sprint}}{\textit{Número de pontos apresentados na Sprint}}$$

- Custo do Ponto: indicador que apresenta o custo do ponto da Sprint. Para o número de horas trabalhadas deverão ser somadas todas as horas, até mesmo as horas das estórias rejeitadas ou não implementadas.

$$\frac{\textit{Quantidade de Horas trabalhadas na Sprint}}{\textit{Número de pontos aceitos}}$$

- Índice de Instabilidade: indicador que apresenta o índice de variação que sofreu uma Sprint quanto ao escopo definido no seu início. Os fatores que influenciam para este índice são: itens planejados, itens incluídos após o início da Sprint, itens que foram retirados da Sprint, e os itens que sofreram alterações de escopo. Assim, este índice pode ser representado por:

$$\frac{\textit{Soma dos itens incluídos, excluídos e modificados após o início da Sprint}}{\textit{Soma dos itens planejados incluídos, excluídos e modificados}}$$

5.6. Ferramentas para gestão de projetos com o Scrum

Para auxiliar o processo de gerenciamento de projetos utilizando o *Scrum*, diversas ferramentas computacionais podem ser utilizadas, a exemplo:

- Redmine
- VersionOne

- *Scrum*Works
- Fire*Scrum*

Importante: o uso de ferramentas para auxiliar a gestão de um projeto *Scrum* pode trazer resultados favoráveis, no entanto é importante tomar cuidado para que o gerenciamento não fique apenas na ferramenta. Ou seja, é recomendável que além da ferramenta, seja utilizado o *Scrum Board*, para que a visibilidade do andamento do projeto não fique comprometida.

5.7. Dificuldades e Desafios

Algumas dificuldades e desafios podem ser encontrados com a implantação do *Scrum* em projetos, como são descritos a seguir:

5.7.1. Falta de comprometimento no projeto

Para o *Scrum*, um dos fatores principais para um bom resultado ao final de um projeto, é o comprometimento de todos no mesmo. Deste modo, deve existir a participação do início ao fim de todos no projeto, devendo realizar alterações nos membros do projeto quando estes faltarem com comprometimento. Pois para obter sucesso, não basta ter apenas um processo bom e bem definido, o projeto deve possuir uma equipe comprometida com resultados. Observe a figura abaixo, onde demonstra a situação de membros comprometidos com o projeto, e alguns apenas envolvidos.

Figura 5.16. Demonstração de Envolvimento x Comprometimento

Uma maneira de resolver esse tipo de situação é trabalhar com projetos de **Metas Compartilhadas**, ou seja, todos devem trabalhar e ajudar uns aos outros para atingir uma meta única, e não as metas individuais, onde um membro não se preocupa com o andamento e desenvolvimento das atividades de outros membros. Desse modo, todos se sentem responsáveis pelo sucesso do projeto.

5.7.2. Product Owner ausente

Existe projeto em que é constatada a ausência e/ou indisponibilidade do *Product Owner* durante o mesmo. Esta situação deve ser resolvida, ou cobrando uma maior participação do PO, explicando que o seu papel é de fundamental importância para um bom gerenciamento

e funcionamento do projeto. Caso o PO continue ausente, o seu papel deve ser atribuído a outra pessoa.

É importante mostrar ao PO que este deve estar disponível para tirar as dúvidas que surgem durante o projeto, e apresentar dados concretos positivos e negativos da sua participação e ausência durante o projeto. Este fato pode ser relatado ao final de cada *Sprint*, durante a reunião de Retrospectiva da *Sprint.*

5.7.3. Scrum Master como membro do Time

Existem situações em que um membro do Time também exerce o papel do *Scrum Master*. É importante salientar que isso pode ser ocasionar em grave problema, pois pode ocorre deste membro não conseguir finalizar todas as suas tarefas para a *Sprint*, gerando assim um impedimento para estas tarefas. Ou este pode ficar ausente de suas atribuições de *Scrum Master*.

O mais viável é que uma pessoa só exerça o papel de *Scrum Master* e nenhum outro papel no projeto. No entanto, existem situações em que isso não é possível. Sendo assim, é mais viável que o membro do Time, que também é *Scrum Master*, só se responsabilize por tarefas que qualquer membro do Time tenha a capacidade de resolver, evitando gerar um novo impedimento quando este estiver atuando como *Scrum Master.*

5.7.4. Falta de maturidade do Time

Para adotar o *Scrum* em um projeto, é importante que o Time possua maturidade, pois as práticas e cerimônias do *Scrum* exigem comprometimento e disciplina para serem realizadas. E em times que falta maturidade, muitas das práticas e cerimônias deixarão de ser realizadas, podendo comprometer os pilares do *Scrum:* transparência, inspeção e adaptação.

Uma das alternativas quando o *Scrum* está sendo adotados em times com pouca maturidade é o uso de práticas isoladas do *Scrum*, ao invés do uso completo do *framework*. Muitas equipes iniciam pelo uso das Reuniões Diárias do Time, uma prática que geram menos desconforto por ser realizada apenas internamente entre os membros do Time.

5.8. Referências

ALLIANCE, A. **Manifesto for Agile Software Development. 2001**. Disponível em: <http://agilemanifesto.org>. Acesso em: março de 2012.

MAGNO, A. F. **Falando sobre Scrum. 2007.** Disponível em: <http://www.visaoagil.com/downloads/biblioteca/axmagnoScrumArmadilhas.pdf>. Acesso em novembro de 2010.

ABRAHAMSSON, P. et al. **Agile software development methods: Review and analysis**. VTT Publications 478, Oulu, Finland, 2002.

BOEHM, B. **View of 20th and 21st Century Software Engineering**. 28th International Conference on Software Engineering, ICSE'06. New York, NY: ACM. 006. p. 12-29.

CAPILUPPI, A. et al. **An Empirical Study of the Evolution of an Agile-Developed Software System**. 29th International Conference on Software Engineering, ICSE'07. Washington: IEEE Computer Society. 2007. p. 511-518.

CHOW, T.; CAOO, D.-B. A survey study of critical success factors in agile software projects. **The Journal of Systems and Software**, 2008. 961-971.

COCKBURN, A.; HIGHSMITH, J. Agile Software Development: The People Factor. **IEEE Computer**, v. 34, n. 11, p. 131-133, 2001.

FERREIRA, C.; COHEN, J. **Agile systems development and stakeholder satisfaction:** a South African empirical study. Annual Research Conference of the South African institute of Computer Scientists and information Technologists on IT Research in Developing Countries: Riding the Wave of Technology, SAICSIT '08. New York: ACM. 2008. p. 48-55.

HO, C.-W. et al. **On Agile Performance Requirements Specification and Testing**. AGILE 2006. Washington: IEEE Computer Society. 2006. p. 47-52.

KNIBERG, H. **Scrum e XP direto das Trincheiras**. Tradução de Tradução de SEA Tecnologia. [S.l.]: C4Media, InfoQ.com, 2007.

KTATA, O.; LÉVESQUE, G. **Agile development:** issues and avenues requiring a substantial enhancement of the business perspective in large projects. Canadian Conference on Computer Science and Software Engineering. Montreal, Quebec, Canada: ACM. 2009. p. 59-66.

RICO, D. F.; SAYANI, H. H.; SONE, S. **The Business Value of Agile Software Methods:** Maximizing ROI with Just-in-Time Processes and Documentation. [S.l.]: J. Ross Publishing, 2009.

SCHWABER, K. **Agile Project Management with Scrum**. Redmond, WA: Microsoft Press, 2004.

SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Pearson Prentice Hakk, 2011.

TEXEIRA, V. S.; DELAMARO, M. E. Geração de Metadados para o Apoio ao Teste Estrutural de Componentes. **Simpósio Brasileiro de Qualidade de Software**, 2008.

WILLIAMS, L. et al. **On the Impact of a Collaborative Pedagogy on African American Millennial Students in Software Engineering**. 29th International Conference on Software Engineering, ICSE'07. Washington: IEEE Computer Society. 2007. p. 667-687.

Capítulo 6

Redes de Petri Estocásticas: Modelagem e Simulação

Rafael Souza, Kádna Camboim, Jean Araujo e Paulo Maciel

Abstract

Petri nets are a mathematical formalism capable of representing digital systems at a high level of abstraction. Its structural aspect is equivalent to a directed graph and bipartite composed of three basic elements known as states, actions and arcs. This chapter presents the history, concepts and applications of Petri nets and its extensions, with emphasis on Stochastic Petri Nets. We also present some dependability measures and their definitions that emphasize the importance of using stochastic Petri nets in the simulation and analysis of different types of systems. A modeling tool for stochastic Petri nets called TimeNet will be explained. Finally, some practical examples will be adopted to serve as a guide for solving exercises.

Resumo

Redes de Petri constituem um formalismo matemático capaz de representar sistemas digitais em um alto nível de abstração. Seu aspecto estrutural equivale a um grafo direcionado e bipartido composto por três elementos básicos conhecidos como estados, ações e arcos. Este capítulo apresenta o histórico, conceitos e aplicações de redes de Petri, bem como suas extensões, com ênfase em Redes de Petri Estocásticas. Também são apresentadas algumas medidas de dependabilidade bem como suas definições que ressalta a importância da utilização das redes de Petri estocásticas na simulação e análise de determinados tipos de sistemas. Uma ferramenta para modelagem de redes de Petri estocásticas chamada TimeNet será explicada. Por fim, serão adotados alguns exemplos práticos que servirão de guia para resolução dos exercícios.

6.1. Introdução

Sistemas concorrentes e distribuídos são alguns dos mais desafiantes sistemas que são desenvolvidos na prática. É importante que atividades de depuração e testes de partes dos sistemas sejam levadas em consideração antes mesmo de sua implantação. Podemos obter isso através da construção de protótipos ou de modelos executáveis, sendo este último, em

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

geral, mais rápido e menos custoso. A simulação destes modelos pode revelar erros ou inconsistências em fases iniciais do processo de desenvolvimento do sistema, permitindo a inferência dos sistemas modelados sem a necessidade de construí-los; também não há a necessidade de perturbá-los, quando seu custo operacional é alto ou os requisitos de segurança impedem ou desaconselham a realização de experimentos, além de não ser necessário correr o risco de danificá-los.

Um modelo é uma representação, muitas vezes em termos matemáticos, do que parecem ser as características importantes do objeto em estudo. Pela manipulação da representação, espera-se que novos conhecimentos sobre o fenómeno modelado e do próprio modelo, seja obtida sem o custo, inconveniência ou perigo de manipular o verdadeiro fenômeno em si. Por exemplo, muitos trabalhos no domínio da energia atômica foram realizados por modelagem devido ao custo e ao perigo de manipulação de materiais radioativos [Peterson 1981]. Os modelos podem ser analíticos ou discretos. Os do tipo analíticos são formados por uma série de equações matemáticas usadas para prever o comportamento do sistema através da atribuição de valores aos parâmetros do modelo. Já os do tipo discretos são representados por uma estrutura matemática ou lógica que pode ser exercitada utilizando-se um computador para simular o comportamento do sistema.

A simulação [Lilja 2000] é um instrumento para modelar características importantes de um sistema. Desta maneira, um simulador pode ser descrito como sendo nada menos que uma forma de teste que pode ser modificada para estudar o efeito causado por este impacto no sistema como um todo. O custo da simulação depende da complexidade do sistema que está sendo observado e do grau de detalhamento que está sendo dado ao simular determinadas funções. A primeira limitação na construção de um simulador é a impossibilidade de se modelar todos os pequenos detalhes do sistema que está sendo estudado, consequentemente simplificações deverão ser supostas para que seja possível a adoção do simulador.

Para utilizar com sucesso a abordagem de modelagem, no entanto, é necessário um conhecimento dos fenômenos modelados e das técnicas de modelagem. Assim, a matemática se desenvolveu como uma ciência em parte por causa de sua utilidade na modelagem de fenômenos em outras ciências [Peterson 1977].

Redes de Petri (PN) é um formalismo caracterizado principalmente pela possibilidade de representar sistemas que exibam características de concorrência e distribuição. Este formalismo apresenta uma notação gráfica que possibilita a apresentação do comportamento dos sistemas modelados, permitindo o uso de seus conceitos de forma intuitiva. Segundo [Peterson 1977] o principal uso das redes de Petri é a modelagem de sistemas de eventos em que é possível que alguns eventos ocorram simultaneamente, mas há restrições sobre o concurso, a precedência, ou a frequência dessas ocorrências.

As redes de Petri foram originalmente desenvolvidas e usadas para o estudo das propriedades qualitativas de sistemas, exibindo características de concorrência e sincronização [Petri 1966]. A PN original não tinha qualquer noção de tempo. Seu objetivo era desenvolver um modelo em que as máquinas de estado fossem capazes de se comunicar. Em relação às aplicações, duas áreas bem-sucedidas são: avaliação de desempenho e protocolos de comunicação. Dentre as áreas de aplicações promissoras, podemos incluir: modelagem e análise de sistemas distribuídos, programas paralelos e concorrentes, sistemas de controle de produção/industrial flexíveis, sistemas de eventos discretos, sistemas com múltiplos processadores de memória, sistemas de computação de fluxo de dados, sistemas de tolerância à falhas, lógica de programação, estruturas e circuitos assíncronos, compilador e sistemas operacionais, sistemas de informações de escritórios e linguagens formais [Murata 1989].

As redes de Petri são assim chamadas em homenagem ao seu criador, Carl Adam Petri [Petri 1966]. Hoje encontramos variações do modelo original sendo usadas em diversas áreas da ciência para ajudar no estudo do comportamento e desempenho de diferentes sistemas.

6.1.1. Histórico

As Redes de Petri (*Petri Nets*) foram inventadas em agosto de 1939 por Carl Adam Petri, quando ele tinha 13 anos, para descrever processos químicos. Vinte e três anos depois, em 1962, ele documentou o trabalho como parte de sua tese de doutorado, intitulada Comunicação com Autômatos (*Kommunikation mit Automaten*), apresentada à Faculdade de Matemática e Física da Universidade Técnica de *Darmstadt* na então Alemanha Ocidental [Petri 1966][Maciel et al. 1996]. A dissertação foi preparada enquanto C. A. Petri trabalhou como cientista na Universidade de *Bonn*, Alemanha, de 1959 a 1962. Este modelo foi baseado nos conceitos de operação assíncrona e concorrente com as peças de um sistema e a percepção de que as relações entre as partes pode ser representada por um gráfico, ou rede [Peterson 1977]. A figura 6.1 apresenta uma fotografia do inventor das redes de Petri já com idade avançada.

Figura 6.1. Carl Adam Petri

O trabalho de Petri atraiu a atenção de A. W. Holt, que mais tarde o levou ao *Information System Theory of Project of Applied Data Research*, nos Estados Unidos [Murata 1989] [Peterson 1977], que em conjunto com outros pesquisadores, desenvolveu muito da teoria, notação e representação das redes de Petri [Maciel et al. 1996].

A década de setenta marcou o desenvolvimento da teoria de redes de Petri e a expansão de seu campo de aplicação. No início daquela década, de 1970 a 1975, o grupo de estrutura da computação do MIT foi o mais ativo na condução da pesquisa sobre redes de Petri e na produção de muitos relatórios e teses [Murata 1989]. Em julho de 1975 houve a conferência Redes de Petri e Métodos Relacionados no MIT, mas nenhum procedimento foi publicado nesse evento. Vale ressaltar que na conferência realizada pelo MIT em 1975, foi a primeira vez em que o nome Rede de Petri foi oficialmente utilizado para se referir ao formalismo proposto por Carl Adam Petri. Em 1979, cerca de 135 pesquisadores de vários

países europeus reuniram-se por duas semanas em Hamburgo, Alemanha Ocidental, para um curso avançado sobre Teoria Geral das Redes de Processos e Sistemas [Murata 1989] [Peterson 1977].

Logo no início da década de 1980, ocorreu o primeiro *Workshop* Europeu em Aplicações e Teoria das Redes de Petri. Desde então, uma série de *workshops* foram realizados todos os anos em diferentes locais da Europa. Contudo, o segundo curso avançado foi realizado na cidade de *Bad Honnef*, no ano de 1986, também na Alemanha, dando seguimento ao primeiro curso realizado em 1979. Em 1989 foi publicado um artigo sobre redes de Petri pelo Professor Tadao Murata [Murata 1989]. Embora este artigo trate primordialmente das redes de *Place-Transition*, ele é um dos artigos mais referenciados sobre o assunto ainda nos dias de hoje.

Somente dez anos depois, em 1996, veio a acontecer o terceiro curso avançado em redes de Petri, desta vez na cidade de *Dagstuhl*, como sempre, na Alemanha, contando com a participação de aproximadamente 80 pessoas, entre os quais 17 palestras foram realizadas, sendo que Carl Adam Petri também visitou o curso por alguns dias [World 2010].

Em 2003, a cidade de *Eichstätt* veio a sediar o quarto curso avançado, em que 24 palestras foram realizadas, contando com a participação de mais de uma centena de pessoas. Coincidentemente, o último evento que segue essa cronologia ocorreu entre os dias 13 e 24 de setembro de 2010, na cidade de *Rostock*, sendo que infelizmente não pôde contar com a presença de C.A. Petri em razão do seu falecimento em julho do mesmo ano, aos 83 anos.

Seguiram diversos outros trabalhos com propostas de alterações ao modelo original, tais como redes com arco inibidor, redes temporizadas determinísticas e estocásticas. Hoje em dia, rede de Petri é considerada uma técnica para especificação de sistemas concorrente já consolidada. Grupos de pesquisa em todo o mundo têm Redes de Petri como tema, estes por sua vez, vem desenvolvendo estudos sobre seus aspectos teóricos e suas aplicações [Maciel et al. 1996].

Para um melhor entendimento da importância das Redes de Petri é necessário que algumas medidas de dependabilidade sejam previamente definidas, assim a sessão a seguir apresenta alguns conceitos importantes.

6.2. Dependabilidade

A avaliação de dependabilidade denota a capacidade que um sistema tem de oferecer um serviço de forma confiável. As medidas de dependabilidade são confiabilidade, disponibilidade, mantenabilidade, performabilidade, segurança, testabilidade, confidencialidade e integridade [Laprie et al. 1992]. Dessa forma, é possível proporcionar um conjunto de serviço para um determinado período de tempo.

A dependabilidade de um sistema computacional pode ser afetada pela ocorrência de eventos de falhas, erros e defeitos, que são mecanismos destrutivos que tentam impedir o correto funcionamento de um sistema em função de uma sucessão de eventos indesejáveis. Estratégias de manutenção são estabelecidas para evitar a ocorrência de tais mecanismos, garantindo a qualidade do serviço oferecido. Dessa forma, estratégias de manutenção têm um impacto fundamental sobre a disponibilidade e a confiabilidade de sistemas [Higgins et al. 2002].

A confiabilidade de um sistema é a probabilidade de que este sistema execute a sua função, de modo satisfatório, sem a ocorrência de falhas, por um determinado período de tempo. De acordo com [Avizienis et al. 2004] confiabilidade de um sistema de computação

é a capacidade de oferecer um serviço que pode justificadamente ser confiável. Um Diagrama de Bloco de Confiabilidade (*Reliability Block Diagram* - RBD) serve para representar o sistema através de blocos de subsistemas ou componentes ligados de acordo com suas funções ou uma relação de confiabilidade e representa as conexões lógicas de componentes necessários para cumprir uma função do sistema especificado [Rausand and Høyland 2004]. RBDs podem ser adotados para representar estruturas em série, paralelo, série-paralelo e paralelo-série para sistemas e redes em geral [Kuo and Zuo 2003].

A disponibilidade de um sistema é a probabilidade de que ele esteja operacional durante um determinado período de tempo, ou tenha sido restaurado após a ocorrência de um evento de falha. Entradas para os modelos de disponibilidade incluem os componentes tempo médio para falha (*Mean Time to Failure* - MTTF) e tempo médio para reparo (*Mean Time to Repair* - MTTR). Os MTTFs são geralmente fornecidos pelo fabricante de hardware. Os MTTRs estão intimamente relacionados com a política de manutenção adotada pela organização. Estas medidas são de grande importância quando se deseja quantificar o efeito das falhas e dos processos de reparo em um sistema e ainda quando se precisa obter tanto o tempo de atividade como o de inatividade. A disponibilidade é obtida por análise de estado estacionário ou simulação, a equação seguinte expressa a relação relativa entre MTTF e MTTR:

$$A = \frac{MTTF}{MTTF + MTTR} \quad (1)$$

Mantenabilidade é a probabilidade de que um sistema seja reparado após a ocorrência de um evento de falha em um determinado período de tempo. A mantenabilidade é descrita pela Equação (2), onde *T* denota o tempo de reparo ou o tempo total de *downtime*. Essa equação representa a mantenabilidade, visto que o tempo de reparo *T* tem uma função de densidade *g(t).*

$$V(t) = P\{T \leq t\} = \int_0^t g(t)dt \quad (2)$$

A performabilidade descreve a degradação do desempenho de sistemas provocada pela ocorrência de eventos de falhas; mesmo em decorrência deles, o sistema continuará funcionando, mas com degradações no nível de desempenho. Segurança é representada pela ausência de consequências catastróficas aos usuários e do ambiente [Avizienis et al. 2004].

De acordo com [Trivedi et al. 2009] a integridade está relacionada a capacidade do sistema impedir modificação ou supressão não autorizadas. Em criptografia e segurança da informação em geral, integridade refere-se à validade dos dados. Desempenho determina o grau em que o sistema ou componente realiza as suas funções designadas dentro das restrições dadas, tais como velocidade, precisão e uso de memória. O desempenho pode ser considerado como um atributo de dependabilidade. Confidencialidade é a capacidade de o sistema evitar a divulgação de informações a terceiros não autorizados, ela deve garantir que as informações sejam acessíveis apenas às pessoas autorizadas.

A disponibilidade e especificação de segurança de um sistema devem incluir os requisitos para os atributos em termos da frequência aceitável e gravidade das falhas de serviço para classes específicas de falhas e uma determinada utilização do ambiente. Um ou mais atributos pode não ser necessários para um dado sistema [Avizienis et al. 2004]. Disponibilidade de um sistema está diretamente relacionada ao tempo de atividade, que

representa o período de tempo em que o sistema está operacional e também ao tempo de inatividade, que é o período de tempo em que o sistema não está operacional devido a ocorrência de um evento de falha ou atividade de reparo.

Para a avaliação de dependabilidade uma abordagem bastante útil é a combinação dos resultados de modelos de diferentes níveis que possuem relação de hierarquia, isso evita o problema de *largeness* que pode incidir em análises que abrangem muitos componentes num mesmo modelo. A avaliação de desempenho de sistemas computacionais consiste de um conjunto de técnicas classificadas como as baseadas em medição e as baseadas em modelagem. As técnicas baseadas em modelagem podem ser classificadas como técnicas analíticas e técnicas baseadas em simulação [Lilja 2000].

A medição de desempenho envolve essencialmente a monitoração do sistema enquanto está sob a ação de uma carga de trabalho. Para adquirir resultados representativos, a carga de trabalho deve ser cuidadosamente selecionada. Essa carga é utilizada nos estudos de desempenho, podendo ser real ou sintética. Embora a carga de trabalho real seja uma boa escolha por representar de forma fiel o sistema, ocasionalmente esta opção não é a desejável. Isso acontece quando o tamanho da carga não é considerável e também quando esses dados receberam muitas perturbações ou até mesmo, por questões de acessibilidade destes. Devido a esses motivos, outros tipos de cargas de trabalho também têm sido usados: *Kernels*, Programas sintéticos e *Benchmarks* [Lilja 2000].

A prestação de serviços tolerante a falhas está relacionada à adição de redundância, pois esta pode ser explorada tanto no tempo ou no espaço, visto que os serviços são fornecidos através de *host* distribuído na rede.

6.3. Classes de redes de Petri

Em algumas fases do processo de desenvolvimento de um sistema, necessitamos representar o sistema em evolução com maior ou menor detalhamento, de maneira compacta. Em outra fase do desenvolvimento pode ser necessário o esclarecimento de determinados aspectos, que em fases anteriores não se faziam necessários [Maciel et al. 1996]. Tendo em vista esses aspectos, a literatura fornece diferentes classificações para as redes de Petri.

A aplicabilidade das Redes de Petri como ferramenta para estudo de sistemas é importante por permitir representação matemática, análise dos modelos e também por fornecer informações úteis sobre a estrutura e o comportamento dinâmico dos sistemas modelados. As aplicações das Redes de Petri podem se dar em muitas áreas (sistemas de manufatura, desenvolvimento de software, sistemas administrativos, entre outros). As redes de Petri são formadas por lugares, transições, arcos e marcas, como apresentado na figura 6.2 (a), (b), (c) e (d) respectivamente.

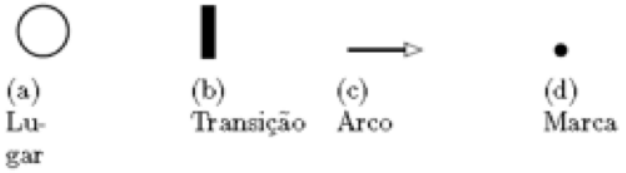


Figura 6.2. Componentes de uma Rede de Petri

Os lugares correspondem às variáveis de estado e às transições às ações ou eventos realizados pelo sistema. A realização de uma ação está associada a algumas pré-condições,

ou seja, existe uma relação entre os lugares e as transições que possibilita ou não a realização de uma determinada ação. Após a realização de uma determinada ação, alguns lugares terão suas informações alteradas, ou seja, a ação criará uma pós-condição. Os arcos representam o fluxo das marcas pela rede de Petri, e as marcas representam o estado em que o sistema se encontra em determinado momento.

Graficamente, os lugares são representados por elipses ou círculos; as transições, por retângulos, os arcos por setas e as marcas por meio de pontos. A figura 6.3 ilustra um exemplo com lugares, arcos e transição.

Figura 6.3. Lugares ligados por arcos e uma transição

Os dois elementos, lugar e transições, são interligados por meio de arcos dirigidos. Os arcos que interligam lugares às transições (Lugar →Transição) correspondem à relação entre as condições verdadeiras (pré-condições), que possibilitam a execução das ações. Os arcos que interligam as transições aos lugares (Transições → Lugar) representam a relação entre as ações e as condições que se tornam verdadeiras com a execução das ações.

As redes se caracterizam pelo tipo de suas marcas, ou seja, suas marcas são do tipo inteiro e não negativo, enquanto as de alto nível possuem marcas de tipos particulares. As redes ordinárias se subdividem em: Redes de Petri Condição-Evento e Redes de Petri *Place/Transition*.

Rede de Petri Condição-Evento ou rede de Petri Elementar é a rede mais elementar dentre todas [Rozenberg and Engelfriet 1998]. Essa rede só permite no máximo um *token* em cada lugar e todos os arcos possuem valor unitário.

Redes de Petri lugar/transição (*Place/Transition*) é a classe mais proeminente e melhor estudada das PNs. Esta classe, às vezes, é chamada simplesmente de rede de Petri. Diferentemente das redes de Petri de sistemas elementares, os lugares (*places*) de uma rede de lugar/transição podem ter qualquer número de *tokens* (fichas). Em contraste com as redes de alto nível, estes *tokens* são indistinguíveis [Desel and Reisig 1998].

O estado de uma *place/transition* é determinado por uma distribuição de *tokens* em seus lugares. Isto é formalmente definido como um mapeamento que atribui a cada lugar um inteiro não negativo. Esse mapeamento é usualmente chamado de marcação. O comportamento básico de uma rede marcada é definido por uma regra de ocorrências locais que determinam as possíveis mudanças elementares na marcação causada por ocorrências de transição. A transição de uma rede marcada é habilitada em uma marcação se cada arco para a transição se origina em um lugar que carrega pelo menos um *token* [Desel and Reisig 1998].

A execução de uma rede de Petri é controlada pela posição e movimento dos *tokens*. Os *tokens* são indicados por pontos pretos e residem em círculos que representam os lugares da rede. Uma rede de Petri com token é uma rede de Petri marcada [Desel and Reisig 1998]. O uso dos tokens mais se assemelha a um jogo de tabuleiro. Estas são as regras: os *tokens* são movidos pelo disparo das transições da rede; a transição deve estar habilitada para o disparo (a transição é ativada quando todos os seus lugares de entrada tem

um *token*); os disparos das transições eliminam os *tokens* dos lugares de entrada gerando novos *tokens* que são depositados nos locais de saída da transição [Peterson 1977]. Esse tipo de rede permite o acúmulo de marcas no mesmo lugar, assim como valores não unitários para os arcos. Contudo, alguns autores fazem distinção entre as redes em que os arcos que têm valores diferentes de um e as redes com arcos de peso igual a um.

A figura 6.4 apresenta um exemplo de uma rede de Petri que possui três lugares (*P0, P1* e *P2*) e duas transições (*T0* e *T1*). No lugar *P0* existem 2 *tokens*, no lugar *P1* inicialmente não há *tokens* e no lugar *P2* existem 5 *tokens*. Os lugares *P0* e *P2* representam as entradas da rede, com o disparo da transição *T0* (a qual está habilitada devido à presença de *tokens* nos lugares de entrada), serão removidos um *token* do lugar *P0* e um *token* do lugar *P2*, que serão depositados no lugar *P1*, habilitando assim a transição T1.

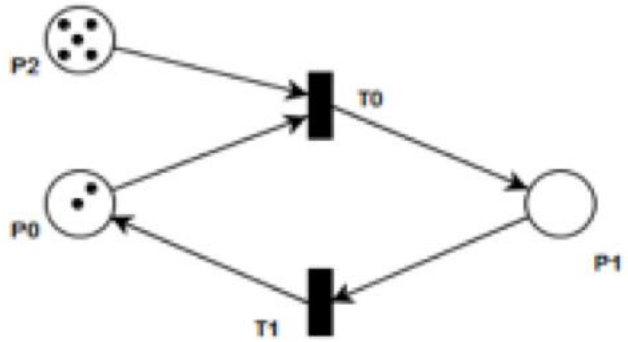


Figura 6.4. Exemplo de uma rede de Petri

Redes de Petri de Alto Nível são caracterizadas pelos tipos de suas marcas, que não são mais elementos do tipo inteiro positivo. São aquelas cujas marcas incorporam alguma semântica, viabilizando sua diferenciação. Esta semântica pode ir desde a atribuição de valores ou cores às marcas, até a adoção de noções de tipos de dados abstratos. Redes não-ordinárias não aumentam o poder de representação de um modelo. Entretanto, elas permitem uma maior clareza e um maior (ou menor) nível de abstração ao modelo. As redes de Petri Temporizadas, Coloridas e Hierárquicas são algumas das principais subdivisões desse tipo de rede.

Os modelos de redes de Petri que foram considerados anteriormente não incluem nenhuma noção de tempo. O conceito de tempo foi propositadamente evitado na obra original de C. A. Petri [Petri 1966], devido ao efeito que o tempo pode ter sobre o comportamento das PNs. De fato, a associação de restrições de tempo com as atividades representadas nos modelos de redes de Petri podem impedir certas transições de disparar, destruindo assim o importante pressuposto que todos os comportamentos possíveis de um sistema real são representados pela estrutura da PN.

Em [Peterson 1981] primeiro livro sobre PNs, aborda a possibilidade de se utilizar redes temporizadas, com a seguinte afirmação: "A adição de informações de tempo pode fornecer uma nova e poderosa ferramenta para as PNs, mas pode não ser possível de forma consistente com a filosofia básica de PNs".

A maioria dos questionamentos iniciais levantados pelos pesquisadores, estão relacionados as propriedades fundamentais dos modelos e sistemas de PN, em suas técnicas de análise e a complexidade computacional associada, na equivalência entre PNs e

outros modelos de computação paralela. Ao lidar com esses problemas, o tempo de fato não é relevante [Marsan et al, 1995] [Balbo 2001].

O disparo de uma transição em um modelo de redes de Petri corresponde ao evento que muda o estado do sistema real. Esta mudança de estado pode ser devido a uma de duas razões: ele deve ser o resultado da verificação de alguma condição lógica do sistema, ou ser induzido pela realização de alguma atividade. Considerando-se o segundo caso, notamos que as transições podem ser usadas para modelar atividades, de modo que a transição habilitada corresponde a períodos de execução de atividade e o disparo da transição corresponde a conclusões da atividade. Assim, o tempo pode ser naturalmente associado às transições [Marsan et al. 1998].

Existem transições que são temporizadas e possivelmente, transições não temporizadas, nesse caso, transições não temporizadas são prioritárias em relação às temporizadas. Desta forma, a propriedade de tempo também pode ser modelada, não somente a estrutura. A principal vantagem de uma rede de Petri temporizada é a relação de tempo, que antes não havia. A relação de tempo altera os estados/marcações alcançáveis. Por este motivo, é muito importante saber utilizar o tempo na rede para que o projeto ou modelo não entre, por exemplo, em um *deadlock* [Maciel et al. 1996].

O tipo de atividade associada com a transição, cuja duração é medida por um contador, depende do sistema modelado real: pode corresponder à execução de uma tarefa por um processador, ou para a transmissão de uma mensagem em uma rede de comunicação, ou ao trabalho realizado em uma parte de uma máquina/ferramenta em um sistema de manufatura. Em fim, várias são as aplicações desse tipo de modelagem para o "mundo real".

Introduzida por *Kurt Jensen* uma *Colored Petri Net* (CPN) ou rede de Petri Colorida tem cada *token* ligado com uma cor, indicando a identidade do *token*. Além disso, cada lugar e cada transição tem em anexo um conjunto de cores. Uma transição pode disparar com relação a cada uma das suas cores [Kurt and Jensen]. Ao disparar uma transição, os *tokens* são removidos dos lugares de entrada e adicionados aos lugares de saída, da mesma forma que em redes de Petri original, exceto que uma dependência funcional é especificada entre a cor do disparo de transição e as cores dos sinais envolvidos. A cor ligada a um *token* pode ser alterada pelo disparo de uma transição e muitas vezes representa um valor complexo de dados. CPNs levam a modelos de rede compactas usando o conceito de cores.

De acordo com [Kurt and Jensen] as redes de Petri Coloridas têm o seu nome porque permitem o uso de *tokens* que transportam valores de dados e podem assim ser distinguidos uns dos outros - em contraste com os sinais das redes de Petri de baixo nível, que por convenção, são desenhadas em preto, ponto "incolor".

O seu principal objetivo é a redução do tamanho do modelo, permitindo que marcas individualizadas (cores) representem diferentes processos ou recursos em uma mesma sub-rede. Inicialmente, os *tokens* das CPNs eram representados por cores ou mesmo por padrões que possibilitam a distinção dos *tokens*. Em trabalhos mais recentes, os *tokens* são representados por estruturas de dados complexas não relacionando cores aos *tokens*, a não ser pelo fato de que estes são distinguíveis [Maciel et al. 1996]. Desta maneira, os *tokens* podem conter informações. Além disso, cada lugar armazena *tokens* de um certo tipo definido e os arcos realizam operações sobre os *tokens*.

CPN é uma linguagem para a modelagem e validação de sistemas em que concorrência, comunicação e sincronização desempenham um papel importante [Jensen et al. 2007]. Um modelo CPN de um sistema é um modelo executável representando os estados do sistema e os eventos (transições), que pode levar o sistema a mudar de estado. A

linguagem CPN torna possível organizar um modelo como um conjunto de módulos e inclui um conceito de tempo para representar o tempo necessário para realizar eventos no sistema modelado.

As redes de Petri Hierárquicas surgiram levando em consideração a modelagem de sistemas de média ou elevada complexidade, em que a utilização de mecanismos de estruturação do modelo que permitem a elevação dos níveis de abstração utilizados é uma necessidade inevitável. A hierarquia tem por objetivo permitir a representação de modelos de sistemas complexos de forma mais compreensível pelo modelador. O conceito de hierarquia em redes de Petri permite o refinamento de lugares de transições.

O refinamento de lugares e transições tem sido utilizado no processo de modelagem hierárquico, contudo o refinamento de elementos vizinhos apresenta dificuldades no controle da consistência do modelo. O refinamento de um elemento pode ser representado como uma sub-rede. Algumas abordagens que tratam hierarquia em redes de Petri possibilitam o refinamento de lugares, transições, lugares e transições e ainda as que não possibilitam o refinamento de elementos adjacentes [Maciel et al. 1996].

Extensões hierárquicas são apenas uma conveniência gráfica que não aumentam o poder de representação das redes de Petri, apenas tornam mais fácil para o usuário a compreensão das redes resultantes, contudo, para a modelagem segura de sistemas de grandes dimensões é necessário o uso de ferramentas que utilizem esses mecanismos.

6.4. Redes de Petri Estocásticas

Esta seção apresenta uma descrição de uma extensão de Redes de Petri Estocástica, que são extensões temporizadas. Detalhes sobre as técnicas de solução e alguns aspectos computacionais são apresentados.

6.4.1. Teoria Estocástica

As redes de Petri permitem descrever uma estrutura lógica de um sistema computacional, não incluindo o conceito de tempo em suas regras. Uma das situações que levou a criação de tempo em redes de Petri estocásticas foi a necessidade de modelar os estados com transições temporais para poder ter uma equivalência entre Cadeias de Markov de Tempo Contínuo (*Continuous Time Markov Chain* - CTMC), tendo estas um papel muito importante na descrição de um sistema complexo e concorrente.

A extensão estocástica, do formalismo Rede de Petri, permite fazer modelagem para análise e otimização de um sistema com característica específica. Isto é porque o paradigma de probabilidade nos obriga a ter uma descrição de um sistema particularmente a fim de que ele seja capaz de formular o experimento probabilístico e de modo a definir variáveis e processo aleatório.

Para um maior embasamento em Goodman [Goodman 1988] encontra-se mais detalhes do assunto, enquanto que uma descrição mais extensa pode ser encontrada em Howard [Howard].

O processo estocástico é uma coleção indexada de variáveis aleatórias Xt, $X \epsilon T$. A variável *t* pode ser discreta ou contínua. A cardinalidade do conjunto de índices *T* indica se o processo é processo discreto ou contínuo [Brzezniak and Zastawniak 2009]. A figura 6.5 e 6.6 ilustram a função amostral que é um conceito de variável aleatória, onde cada instante tem-se uma variável aleatória diferente.

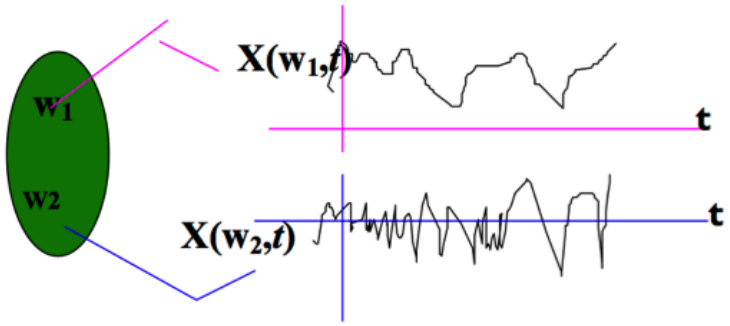


Figura 6.5. Função Amostral

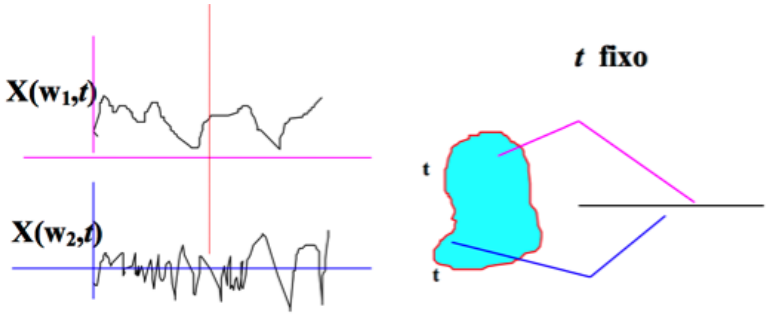


Figura 6.6. Segundo exemplo de Função Amostral

As probabilidades apresentam técnicas mais eficazes para situações que envolvem incertezas ou aleatoriedade, este campo da matemática fornece mecanismo para estas situações. Uma vez que uma situação problema tenha sido formulada como um experimento probabilístico, em seguida, a teoria da probabilidade pode ser aplicada [Bertsekas and Tsitsiklis 2002].

6.4.2. Redes de Petri Estocásticas

6.4.2.1. Conceito

Rede de Petri Estocástica (*Stochastic Petri Net* - SPN) é uma extensão do formalismo Rede de Petri que pode ser definida como uma ferramenta para modelagem e avaliação de dependabilidade e desempenho de sistema envolvendo concorrência e sincronismo que foram adicionadas variáveis aleatórias para representar a duração das atividades, ou atraso até o evento, permitindo obter métricas de confiabilidade e apresentando um mapeamento muito simples entre SPN e eventos de processo de Cadeias de Makov [Balbo 2001].

O conceito estocástico teve inicio na década de 70 e vários pesquisadores contribuíram acrescentando diferentes características estocásticas, a qual foram criadas extensões estocástica. Entre os pesquisadores podemos citar Noe e Nutt [Noe and Nutt. 1976], Merlin e Farber [Merlin and Farber. 1976] e Zuberek [Zuberek 1980] que tratam de modelagem do comportamento de sistemas computacionais dinâmicos.

Na década de 80 [Molloy 1982] definiu na sua tese de doutorado o modelo de rede de Petri estocástico, uma melhoria capaz de especificar sistemas, também apresentou uma forma de modelar os sistemas utilizando análise probabilística. Entre as melhorias está a

temporização, implantadas em cada transição, a qual foi o foco principal para a criação do processo estocástico.

As transições possuem tempo, disparo atômico e retardo de transição através de variáveis aleatórias distribuídas exponencialmente. O comportamento das variáveis pode ser descrito através de um processo estocástico, juntamente com a utilização de *memorelles*, que é a falta de memória, dessa forma, tornam-se isomórfica as cadeias de Makov. Assim, é possível medir o desempenho e a dependabilidade do sistema. A modelagem básica de uma transição temporizada é apresentada na figura 6.7. A atividade representada pela transição *t1* está habilitada devido à presença de marcações (*tokens*) nos lugares de entrada *P1* e *P2*, após o disparo os *tokens* são inseridos no lugar de saída *P3*.

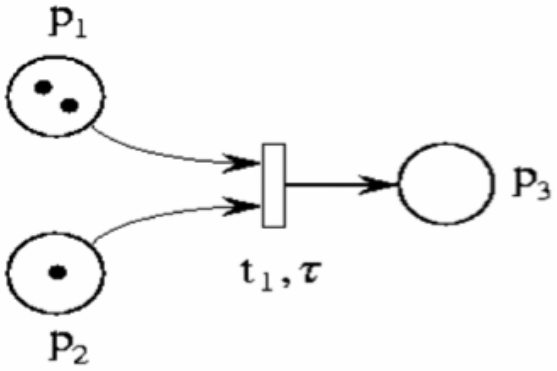


Figura 6.7. Rede de Petri com transição temporizada

Esta característica de tempo nas transições permite diferentes tipos de funcionamento para o disparo, podemos citar, o disparo em três fases ou os disparo atômico. No caso do disparo em três fases se distingue como:

Os *tokens* são eliminados quando a transição habilitada;
Espera um tempo;
Os *tokens* são aceitos pelo *token* de destino.

Já o disparo atômico tem a seguinte característica:

- Os *tokens* permanecem no lugar de entrada até ser disparada a transição;
- A eliminação/consumo dos *tokens* de entrada se dá pela aceitação do lugar destino.

Dessa forma, é possível a construção de modelos realistas para avaliar sistemas a eventos discretos para simulação. Portanto, redes estocásticas tem associado a sua transição o tempo que é exponencialmente distribuído, o que facilita a construção de processos Markovianos equivalente e permite ainda a análise do comportamento da rede. Como elas produzem cadeias de Markov que podem ser resolvidas com o padrão de técnicas numéricas, métodos de solução mais avançados com base em matriz geométrica podem ser utilizados.

Redes de Petri estocásticas são abordadas neste capítulo para representar ações de natureza aleatória, devido, por exemplo, a ocorrência de falhas ou estratégias de serviço

aleatórias. Sendo esta uma extensão dentre as varias extensões estocásticas. Para maiores informações sobre redes de Petri estocásticas generalizada consulte [Marsan et al. 1998] e [Ajmone Marsan et al. 1984]. Para redes de Petri estocástica de alto nível, maiores detalhes podem ser obtidos em [F.J.W.Simons 1978]. Em [K.Lautenbach 1978] é dado um detalhamento a parte numérica, dentro dos quais se encontram as predicações/transições. Mais explicações sobre as redes de Petri colorida pode ser vista em [Jensen et al. 2007], [Kurt and Jensen] e [Jensen 1997].

Este formalismo pode ser usado em várias áreas como sistemas complexos, sistemas de automação, *cloud computing*, planejamento de infraestrutura de *data center*, para encontrar requisitos como desempenho, disponibilidade, confiabilidade e manutenção de um sistema.

A análise SPN, bem como a Makoviana não utilizam da memória do passado (*memoreless*), isto é, um evento que produzir um disparo na transição *Mx* em *My*, fará que a transição sensibilizada por *Mx* antes do disparo seja igual aquela que as transições sofreriam se viessem a ser sensibilizada por *My*.

Portanto, as SPN são redes com distribuições geométricas e exponenciais, cuja duração da sensibilização da rede é associada à transição. Diante do exposto pode-se encontrar métricas de dependabilidade (*dependability*) e desempenho (*performance*). Todas as métricas exposta pela técnica de dependabilidade podem ser criadas e avaliadas pelos métodos formais de SPN.

6.4.2.2. Definição

Redes de Petri são uma ferramenta clássica para a modelagem e análise de sistemas de eventos discretos que são demasiado complexo para ser descrito por autômatos ou modelos de filas [Reisig, 1985)]. Atrasos estocásticos e escolhas probabilística são aspectos essenciais para um modelo de avaliação de desempenho. Uma definição SPN estendida é dada em [German, 2000]:

A SPN = (P, T, I, O, H, Õ, G, M0 , Atts) é uma rede de Petri estocástica, onde:

1. P = (p1, p2, … , pn) é o conjunto de lugares,
2. T = (t1, t2; …, tn) é o conjunto de transições,
3. I $\in (N^n \to N)^n \; x \; n$é uma matriz de múltiplas marcações dependente de arcos entrada, onde entradas de i_{jk} da a multiplicidade de arcos (possivelmente marcação dependente) de arcos de entrada do lugar p_j para a transição A $\subseteq (Px\,T) \cup (T)$,
4. $0 \in (N^n \to N)^{NxN}$ é uma matriz de marcações dependentes de múltiplos arcos de saída, onde entradas de O especifica a possibilidade de marcação dependente de múltiplos arcos de saída a partir da transição para o lugar
5. $H \in (N^N \to N)^{N\,x\,N}$: é a matriz de marcações dependente que descreve a multiplicidade de arcos inibidores, onde entradas h_{jk}de H retorna a possibilidade de marcação dependente de múltiplos arcos inibidores a partir do lugar p_j para a transição t_k,
6. $\prod \in N^m$:é um vetor que atribui um nível de prioridade para cada transição,

7. G $\in(N^n \rightarrow (true, false))^m$ é um vetor que atribui uma condição de guarda relacionada

ao lugares marcados para cada transição. (Na presença de um arco de inibidor, uma transição é disparada se cada lugar de entrada ligado por um arco normal tem um número de tokens igual ao peso do arco, e se cada lugar de entrada ligado por um arco inibidor não tem nenhum token.),

- $M_0 \in N^n$ é um vetor que contém a marcação inicial para cada lugar (estado inicial),
- $Atts = (Dist, W, Markdop, Policy, Concurrency)^m$ compreende o conjunto de atributos

para transições, onde:

- $Dist \in N^m \rightarrow F$ é uma possibilidade de disparo de marcação dependente da probabilidade de função de distribuição (esta distribuição pode ser dependente de marcação)

(o domínio de é [0;¥]),

- $W{:}T \rightarrow I\,R$ é a função de peso, que representa o peso (W_t) de transições imediatas e a taxa λ_t de transições temporizadas, onde:

$$W(t) = \begin{cases} W_t & \geq 0, \text{ se t é uma transição imediata;} \\ \lambda_t & <0, \text{ caso contrario.} \end{cases}$$

- $Markdep \in \{constant, enabdep\}$ onde o disparo da transição pode ser distribuição temporizada independente de marcação (*constant*) ou dependente de marcação (*enabdep*) a distribuição depende da condição atual de habilitação),
- $Policy \in \{prd, prs\}$, é a política de preempção que significa que quando uma condição preemptiva torna-se habilitada novamente o tempo de disparo anterior é perdido; O prs (*preemptive resume*), em que tempo de disparo relacionado a uma trasição preemptiva é retomado quando a transição se torna ativa novamente),
- $Concurrency \in \{ss, is\}$; isg o grau de concorrência de transições, onde representa a semântica finity server e retrata a semântica de infinity server no mesmo sentido como em modelos de fila.

Em muitas circunstâncias, pode ser mais adequado representar a marcação inicial como um mapeamento do conjunto de lugares para números naturais onde $m_0(pi)$ indica a marcação inicial do lugar $pi\,x\,m\,(pi)$ indica uma marcação alcançável (estado alcançável) do lugar pi.

Rede de Petri Estocástica (SPN) introduz o conceito de tempo no formalismo de Redes de Petri permitindo a descrição de um comportamento dinâmico de sistemas na modelagem de dependabilidade [Marsan et al.]. SPNs são definidas assumindo que todas as transições são temporizadas e que o atraso no disparo das transições é associado a uma variável aleatória distribuída exponencialmente.

A transição temporizada é constituída de um atraso que provoca um retardo exponencial no disparo da transição juntamente com a utilização de *memoreless* (falta de memória), o que torna isomórfica as cadeias de Makov e ainda permite medir a dependabilidade do sistema.

As transições em SPNs podem ser imediatas com tempos associados que são exponencialmente distribuídos e temporizadas onde o tempo associado é zero. Nas transições temporizadas o período de habilitação corresponde ao período de execução da atividade e o disparo corresponde ao término da atividade. Diferentes níveis de prioridade podem ser atribuídos às transições. A prioridade de disparo das transições imediatas é superior à das transições temporizadas. As prioridades podem solucionar situações de confusão [Marsana et al.]. As probabilidades de disparo associadas às transições imediatas podem solucionar situações de conflito [Balbo, Marsana et al.].

Os modelos SPN possuem dois tipos de estados (marcações), os estados *vanish* e os estados *tangible*. Os estados voláteis (*vanish*) são criados em decorrência da marcação dos lugares que são pré-condições de habilitação de uma transição imediata. O termo *vanish* é usado porque as marcações chegam a esses lugares e são instantaneamente consumidas. O tempo de permanência das marcações nesses lugares é zero. Os estados tangíveis (*tangible*) são criados em decorrência da marcação dos lugares que são pré-condições de habilitação de uma transição temporizada [Marsana et al.].

Redes de Petri estocásticas são marcadas com um número finito de lugares e transições, são isomórficas as cadeias de Markov [Murata]. O isomorfismo de um modelo SPN com uma cadeia de Markov é obtido a partir do gráfico de alcançabilidade reduzido, que é dado através da eliminação dos estados voláteis e rótulo dos arcos com as taxas das transições temporizadas e pesos das transições imediatas.

O formalismo das Redes de Petri é utilizado em sistemas que possam apresentar atividades assíncronas, concorrentes e não-determinísticas, além de conflitos. A alocação de recursos, sistemas operacionais, redes de filas, entre outros, são exemplos de áreas de estudo em que se aplicam a modelagem de redes de Petri estocásticas. Redes de Petri estocástica (SPNs) é uma ferramenta gráfica cujo propósito é a abstração de um sistema real, um formalismo do fluxo de dados do sistema modelado [Haverkort et al.].

O formalismo estocástico apresenta vários modelos. Este formalismo pode ser usado em várias áreas de sistemas complexos como sistemas de automação, sistemas aéreos, sistemas de tempo real, dentre outros, para encontrar métricas importantes, tais como confiabilidade, disponibilidade, mantenabilidade, etc.

A vantagem oferecida para uso de modelos estocásticos está na possibilidade de adicionar valores para determinados parâmetros de entrada a fim de obter valores parâmetros de saída, bem como a simplicidade oferecida por ferramentas de interface amigável para modelagem de sistemas, como exemplo a ferramenta TimeNet.

6.4.2.3. Regras de habilitação e disparo

A funcionalidade para a regra de habilitação e disparo se dá pela execução de um disparo da transição, de forma que é necessário que a transição esteja habilitada para isto, e que tenha pelo menos um *token* no lugar que o antecede a transição.

Para ser efetuado o disparo é necessário que as regras de condições de habilitação sejam verdadeiras, caso contrário não ocorrerá o disparado. Ao ser disparado uma transição *tokens* do lugar de entrada são removidos e colocados no lugar de saída, para exemplificar melhor a situação de disparo a figura 6.8 apresenta o funcionamento de uma rede de telefonia. Após o disparo da transição reserva para *t*, um *token* será removido do lugar de entrada *P1*, consequentemente será depositado no lugar chamado *duração* que habilitará a transição *t*. As transições serão disparadas após decorrido o tempo de espera.

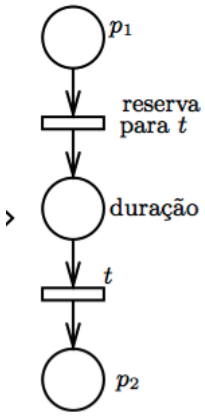
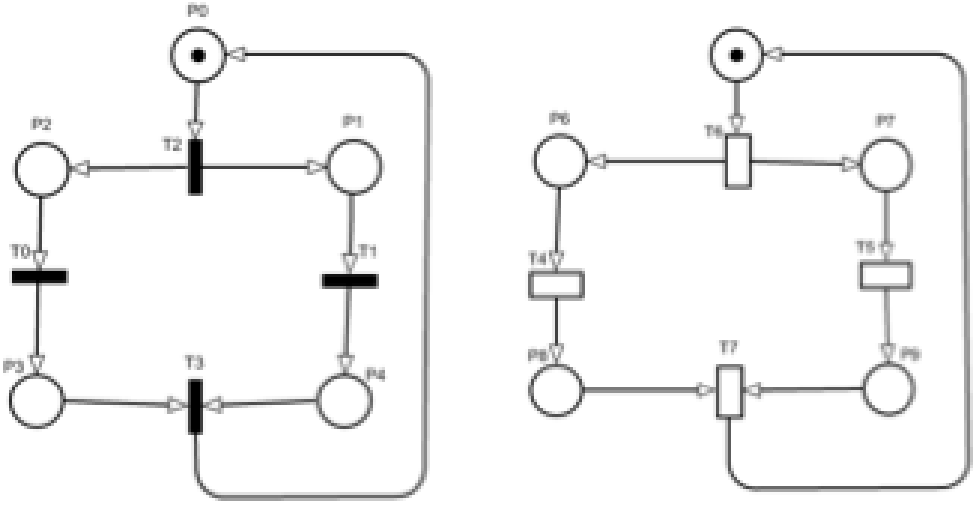


Figura 6.8. Rede de telefonia com transições temporizadas

A figura 6.9a apresenta uma rede de Petri estocástica e a figura 6.9b apresenta o mesmo exemplo de rede de Petri sem o conceito de tempo. Como mostrado na figura 6.9a , o *token* é deslocado do lugar P0, removido e disparado pela transição imediata T2, produzindo uma nova marcação nos lugares P1 e P2. Todas as transições tem associado a si tempos que são exponencialmente distribuídos, o que pode modificar a distribuição dos *tokens* nos lugares e produzir distribuição diferenciada. Os *tokens* são considerados entidades indivisíveis, ou seja, um disparo de uma transição pode desativar outros possíveis disparo, e remover *tokens* que poderia ser consumido por outra transição que o alimenta pelo mesmo lugar de entrada. Ainda observando a figura 6.9a, pode-se notar que após o disparo da transição T2, os lugares P1 e P2 receberam um *token* cada. Por consequência, as transições T0 e T1 passaram a estar habilitadas para o disparo. O disparo da transição T0 coloca o *token* no lugar P3 e habilita a transição T3, o disparo da transição T1 coloca o *token* no lugar P4 e também permite habilitar o disparo da transição T3. Neste momento a transição T3 está habilitada, ou seja, pode ser disparada a qualquer momento. A seleção do disparo é não determinista e o disparo de uma transição desativa automaticamente o disparo da outra.

A figura 6.9b apresenta a mesma rede ilustrada na figura 6.9a, com a diferença de que as transições não são imediatas e sim exponenciais. O lugar inicialmente marcado habilita a transição T6. Após o disparo da transição T6, *tokens* são inseridos nos lugares P6 e P7. Assim, as transições T4 e T5 tornam-se habilitadas. Um determinado *dalay* (atraso) está associado às transições, cada transição pode ter um *delay* diferente. Após o disparo da transição T4, um *token* é colocado no lugar P8, isso habilita a transição T7. Após o disparo da transição T5, um *token* é colocado no lugar P9 o que habilita a transição T7.



a) **b)**

Figura 6.9. Diferenças entre os modelos

6.5. Visão Geral da Ferramenta TimeNet

A ferramenta TimeNET foi desenvolvida pela instituição de ensino Alemã *Technische Universitat Berlin*, a iniciativa para o desenvolvimento desta ferramenta foi para modelar Redes de Petri temporizadas não Markovianas. A criação do TimeNET partiu de uma união de ferramentas existente anteriormente na universidade de *Technische*. Essas ferramentas que deram origem ao TimeNET foram DSPNexpress e GreatSPN, cuja a influência maior se deu por parte da ferramenta GreatSPN. A ferramenta TimeNET será descrita nesta seção, para qual foi utilizada a versão 4.0, lançada em 2007. A figura 6.10 ilustra a interface gráfica do TimeNET com algumas redes já modeladas [German et al. 1995].

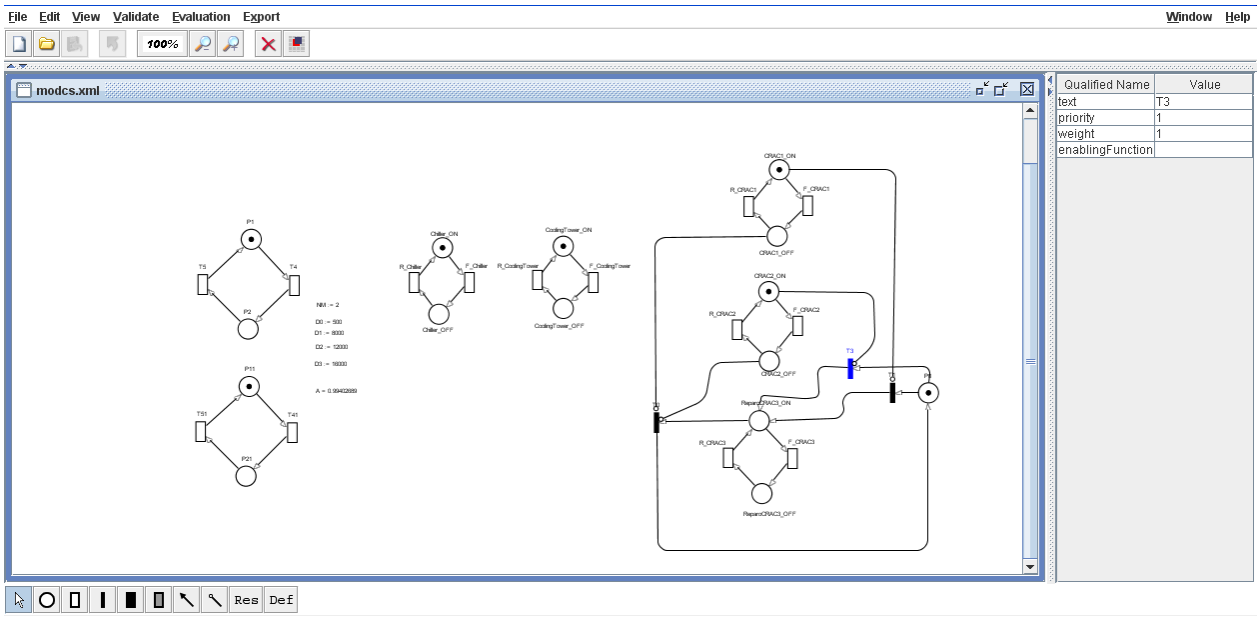


Figura 6.10. Interface do TimeNET 4.0

Esta ferramenta oferece diversas vantagens em relação à outras ferramentas existentes, sendo a principal delas a implementação dos algoritmos de análise numérica de redes de Petri estocásticas o que permite a criação de transições com tempo de disparo determinístico, distribuídos exponencialmente, também com classe especial de distribuição e a distribuição expolinomial. Mais detalhes pode ser encontrado em [Zimmermann and Freiheit 1998] .

A grande aceitação da ferramenta TimeNET por instituições acadêmicas se deu por conta da facilidade e robustez na modelagem e análise de redes de Petri, entre as várias distribuições aceitas estão:

- Redes de Petri Coloridas Hierárquicas: Esta classe de redes de Petri pode conter *tokens* sem tipo, que não armazenam valores, bem como *tokens* coloridos que possuem tipos, portanto, são capazes de armazenar valores.
- Redes de Petri Estocásticas Fluídas: Este formalismo é uma extensão da classe de redes de Petri GSPN, no qual não só se conhece lugares que contém determinado número de *tokens*, como também permite que um ou mais lugares tenham uma quantidade contínua de fluído, que é representado por um número real não negativo, em vez de apenas um número discreto de *tokens*. Este "fluído" pode ser transferido através arcos fluídos em um fluxo contínuo, tanto quanto a transição em questão permitir. Também é permitido que uma determinada quantidade de fluído seja depositada ou removida de uma vez. São bastante úteis na avaliação de sistemas que envolvem componentes como tempo, líquidos, temperaturas, etc.
- Redes de Petri Determinísticas e Estocásticas Estendidas: Esta classe de redes de Petri consiste em uma extensão à classe GSPN. Nela, são permitidas transições com distribuições de disparo imediatas (zero), exponencialmente distribuídas, determinísticas, ou pertencer a classe de distribuições expolinomial, já citada anteriormente.

A figura 6.11, ilustra as etapas para o processo de simulação, tendo inicialmente a criação do modelo de SPN, iniciando em uma segunda etapa a distribuição de processo *slave*, os modelos são executados a nível de kernel, onde os vários arquivos *slave* executa a probabilidade de funcionamento do sistema, ao termino da probabilidade é levado uma amostra de medição, sobre esta amostra é feita a análise estatística, obtendo assim o resultado.

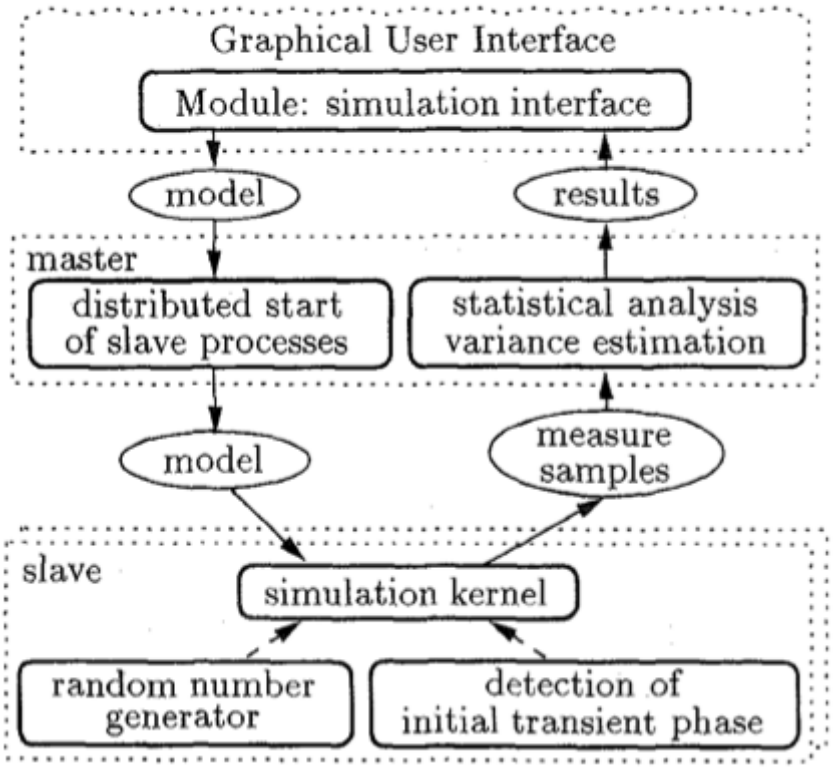


Figura 6.11. Simulação Distribuída

A figura 6.12, apresenta uma análise através do módulo de análise. Os dados são seguidos em criação do modelo, ao ser executado é gerado o grafo de alcançabilidade, o grafo tanto pode gerar os espaço de estado exponencial, como não exponencial. Sendo exponencial os dados são gerados através da escolha do botão *direct*, que gera uma matriz C, o processo pode ir tanto para uma matriz P, como conversão, isto depende do processo. Gerando uma conversão, as probabilidades de estado são criadas, então são computados os resultados e levados a interface gráfica para o usuário. Se a opção escolhida não for o *direct*, se da início ao processo onde são criados subprocesso em cadeia de Markov randomicamente para a matriz P, após esta na matriz um *solve* da equação linear do sistema faz execução, gerando uma solução em cadeia de Markov, seguidamente é gerado a conversão dos dados, em probabilidades de estados, e então computado o resultado, e também levado ao usuário através da interface gráfica.

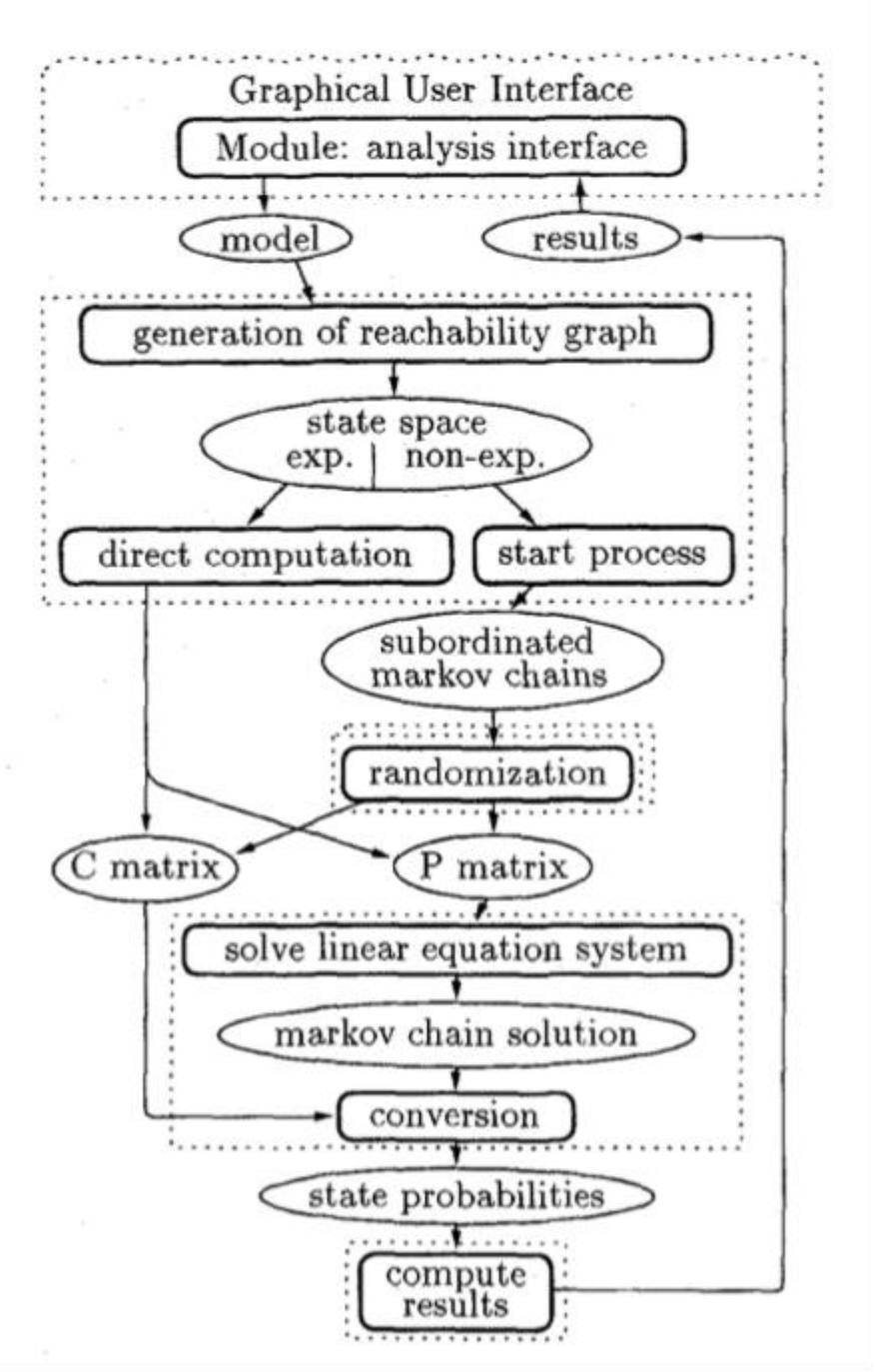


Figura 6.12. Módulo de Análise

A interface gráfica permite criar através do *netclass*, os modelos SCPN e eDSPN em que os formatos são definidos pelo XML. A figura 6.13 apresenta a interface do TimeNET para criação de modelos eDSPN para modelagem de Redes de Petri determinísticas.

A figura 6.14 apresenta a interface que permite a criação de modelos SCPN para modelagem de sistemas por redes de Petri estocásticas colorida.

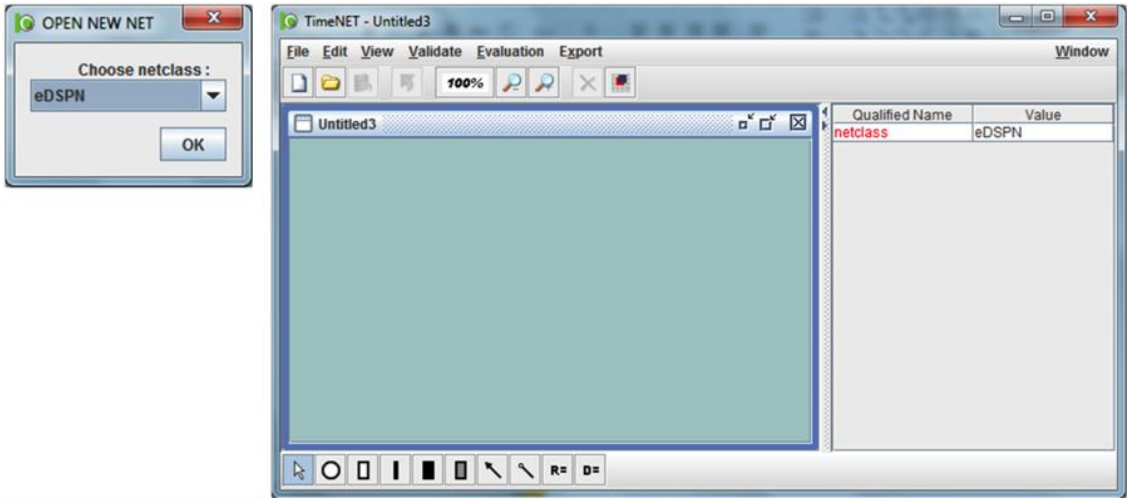


Figura 6.13. Escolha do modelo eDSPN

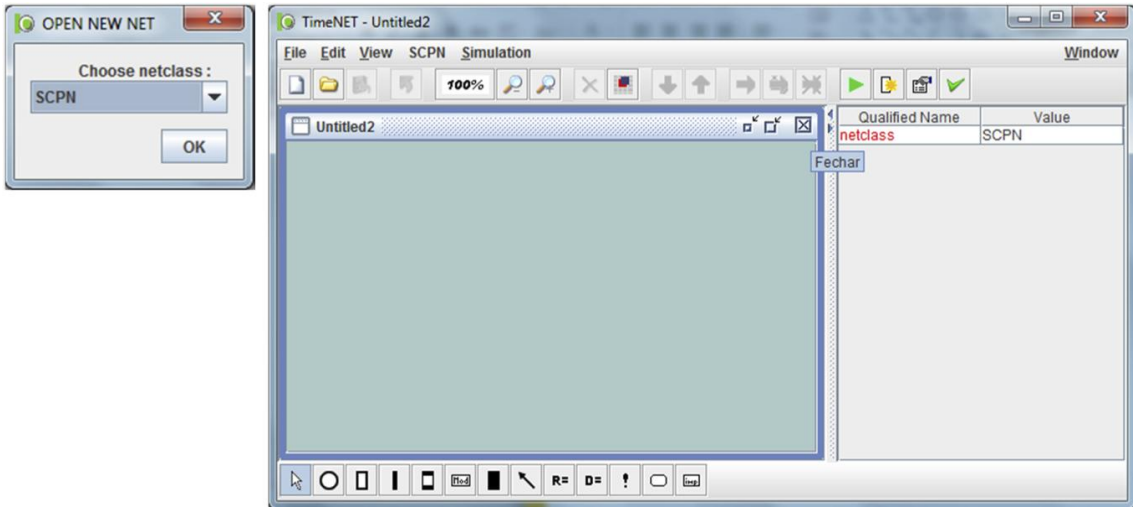
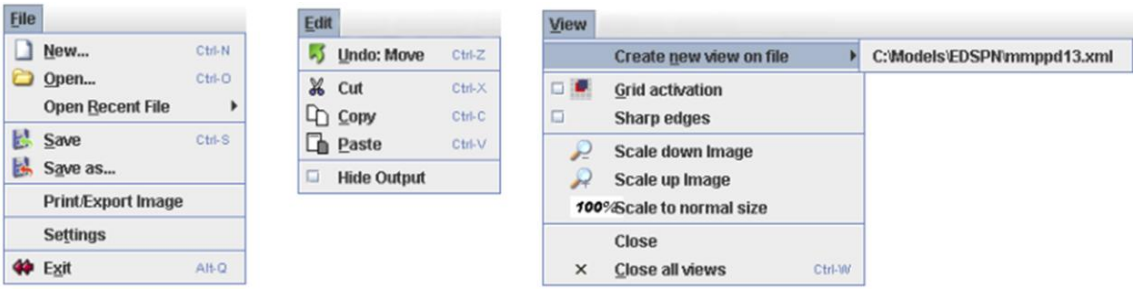


Figura 6.14. Escolha do modelo SCPN

A figura 6.15a apresenta uma sequência de *menus* da interface do TimeNET. Pode-se observar os *menus File*, *Edit* e *View*. Cada um deles apresenta uma série de opções que podem ser escolhidas para realização de determinadas tarefas. No *menu File* existe opções para a criação de um novo modelo, representado por NEW. é possível abrir um projeto pela opção *Open* ou um projeto que já foi aberto recentemente pela opção *Open Recent File*. Para salvar os arquivos modelados pode-se escolher as opções *Save* ou *Save as*. Para imprimir ou exportar uma figura na formato TN ou PNML use a opção *Print/Export Image*. Settings é usada para configurar a interface. Através de *Exit* um projeto é fechado. A figura 6.15b apresenta as opções do *menu Edit*, são elas: *Undo: Move* para voltar um passo, *cut*, *copy* e *paste* para recortar, copiar e colar, respectivamente. A figura 6.15c apresenta o *menu View* com as opções de criar uma nova janela no editor com o mesmo arquivo (*create new view on file*), exibir os modelos em grades, aumentar a imagem na tela ou exibir em escala normal e ainda fechar as visualizações.



a) b) c)

Figura 6.15. Menus File, Edit e View do TimeNET com suas opções

A figura 6.16 mostra os botões de atalhos da interface do timeNET para os comando de criar um novo projeto, abrir, salvar, voltar uma etapa anterior, escala normal de *zoom*, diminuir *zoom*, aumentar zoom, excluir alguma parte do modelo ou o modelo todo e por fim alternar sua visualização.

Figura 6.16. Os botões comando

A figura 6.17 apresenta os botões objeto para criação das redes de Petri estocástica. O primeiro botão representado como seta serve para utilização normal do *mouse*. O segundo botão, representado por uma elipse é utilizado para criação dos lugar. O terceiro botão permite inserir transições do tipo exponencial. O quarto botão para transições imediatas. O quinto para transições determinísticas e o sexto para transições geral. O sétimo botão permite criar arcos que ligam os lugares as transições e vice-versa. O oitavo para criar um arco inibidor. O nono deve ser utilizado para criar parâmetros que obtém resultados a partir da execução do modelo. Por fim, o décimo botão permite inserir parâmetros cujos valores são previamente definidos.

Figura 6.17. Botões objeto

A figura 6.18 apresenta as opções que podem ser realizadas clicando com o botão direito do *mouse* em cima de um objeto selecionado (transição). É possível apagar ou rotacionar para direita ou esquerda. A mesma ação pode ser escolhida para manipular os objetos (arcos e lugares) de uma SPN, porém apenas a opção de apagar (*delete*) será mostrada.

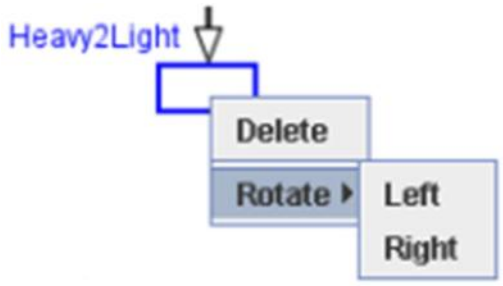


Figura 6.18. Selecionando opções de um objeto

A figura 6.19 mostra o menu *validate* com as opções *Estimete Statespace* para estimar o espaço de estados, *Traps*, *Siphons*, *Check Structure* para verificar a estrutura do modelo, falhas podem ser identificadas, e *Tokengame* para ativar os estados do modelo e verificar a lógica funcional.

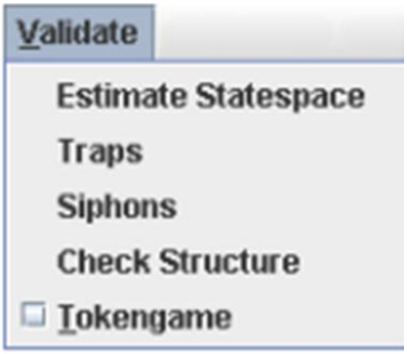


Figura 6.19. Menu Validate e suas opções

Toda as avaliações são feitas através do menu de *Evaluation*, como mostra a figura 6.20. A partir deste menu temos importantes opções para avaliações, como a de *Stationary Analysis* para realização de análise estacionária, *Stationary Approximation* para aproximação estacionaria, *Stationary Simulation* para simulação estacionaria, *Transient Analysis* para análise transiente e *Transient Simulation* para simulação transiente.

Para realização da análise estacionária uma nova janela é aberta para configuração de determinados parâmetros que devem ser definidos pelo usuário, como por exemplo o número máximo de interações, precisão aritmética, modo do sistema linear, dentre outros. Observe a figura 6.21 que mostra todas as opções disponíveis.

Figura 6.20. Menu *Evaluation* e suas opções

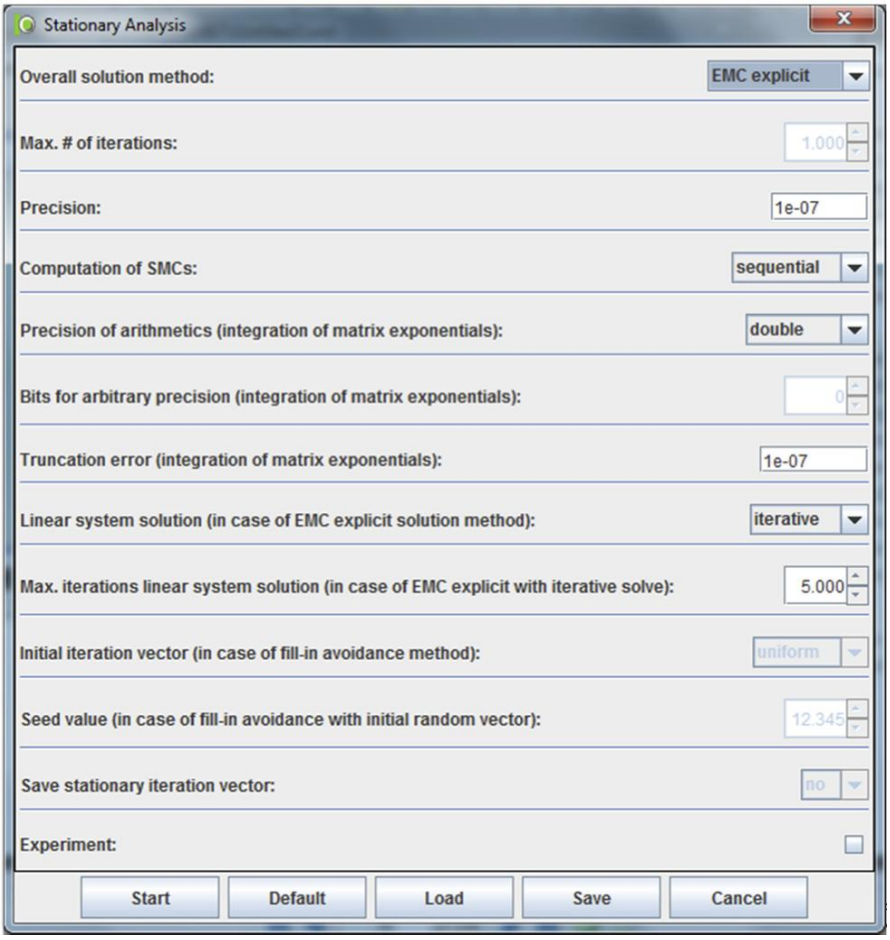


Figura 6.21. Janela de configuração para Análise Estacionária

Caos seja feita a escolha para realização de simulação estacionária, uma nova janela é aberta para configuração de determinados parâmetros que devem ser definidos pelo usuário, como por exemplo o número máximo de amostras, dentre outros. Observe a figura 6.22.

A figura 6.23 mostra as opções que devem ser preenchidas para realização de experimento.

As figuras 6.24 e 6.25 apresentam as janelas para preenchimento de algumas opções quando se é escolhida análise transiente e simulação transiente, respectivamente. O preenchimento das opções de forma bem definida permite um melhor nível de abstração de sistemas reais.

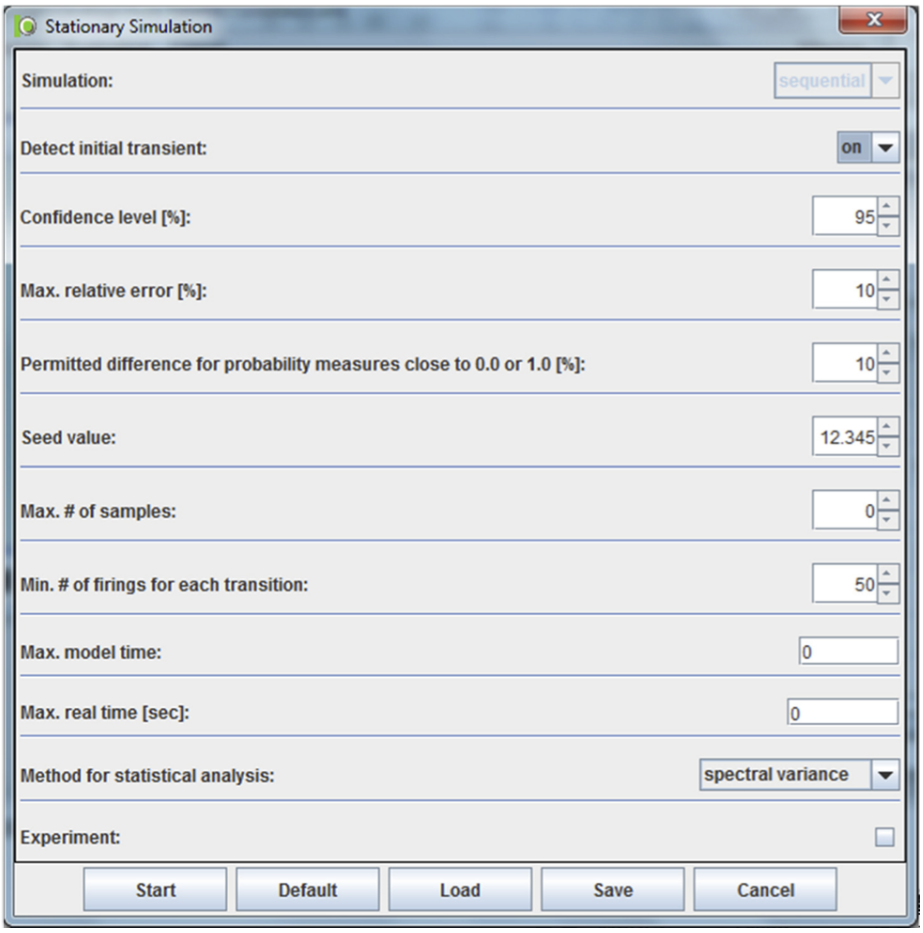


Figura 6.22. Janela de configuração para Simulação Estacionária

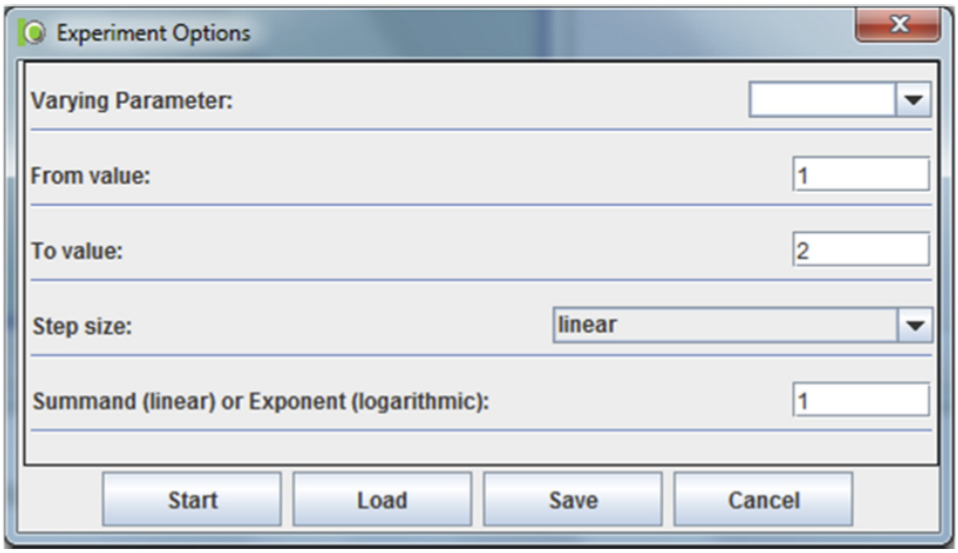


Figura 6.23. Opções para realização de experimento

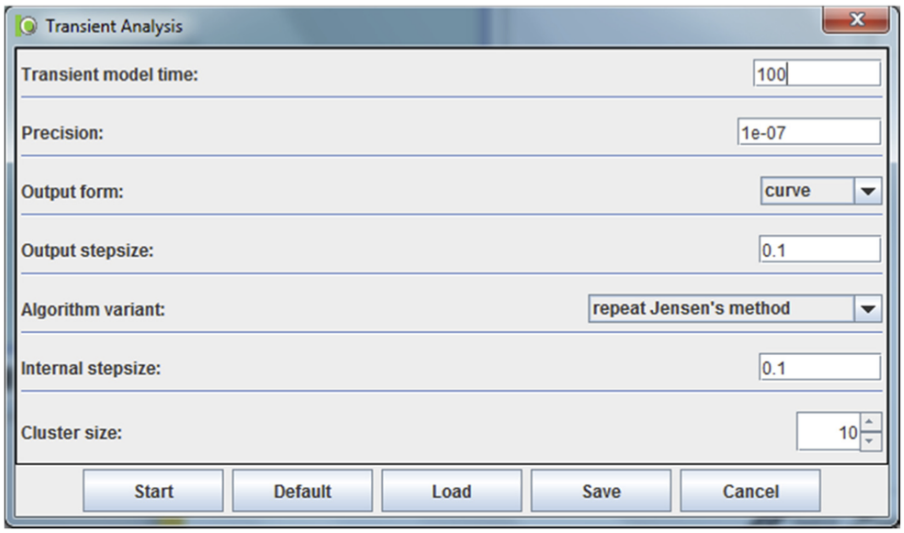


Figura 6.24. Opções para realização de Análise Transiente

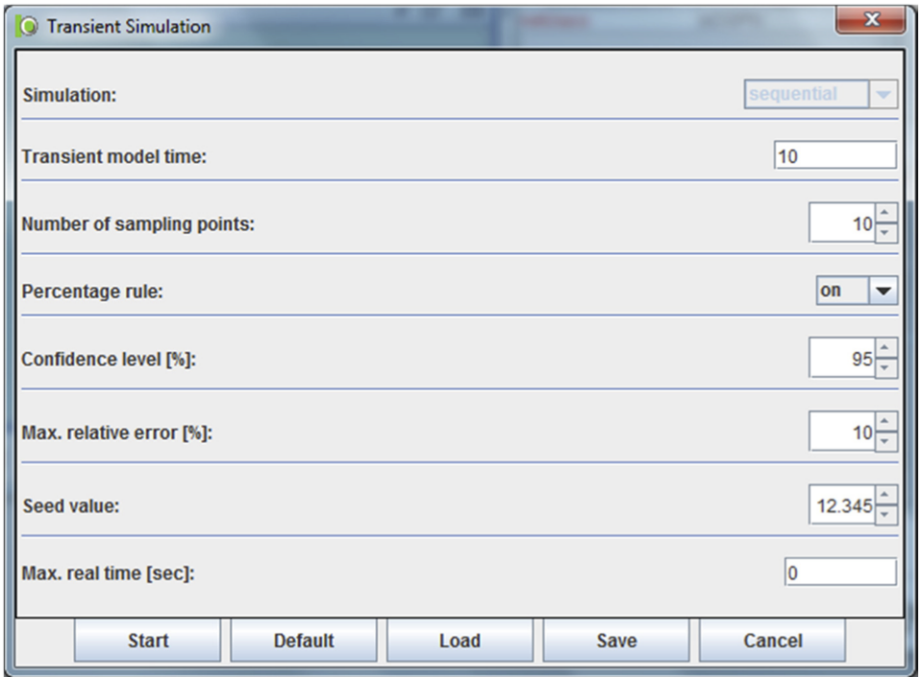


Figura 6.25. Opções para realização de Simulação Transiente

Para criação de uma rede de Petri estocástica, inicie o TimeNET, escolha o modelo SCPN, utilize os botões objetos para criar os lugares, transições e arcos de ligação, em seguida defina o número de marcas para os lugares de entrada e salve o modelo. A partir daí você pode verificar a estrutura do modelo, a lógica funcional e fazer avaliação que se enquadrem com as necessidades do sistema modelado.

Esta seção apresentou alguns detalhes importantes da ferramenta TimeNET bem como instruções inicias para criação de um modelo.

6.6. Exemplos

Esta seção apresenta exemplos com diferentes modelos de redes de Petri estocástica. Para um melhor embasamento para o leitor, serão mostrados cinco exemplos, são eles:

1. Para um sistema com multiprocessador simples que necessita/solicita ter acesso á mesma memoria ou memoria comum ao sistema. Um processador executa localmente por algum tempo, com duração média de 1= L, e solicita o acesso á memoria comum assumindo que para ganhar o acesso tem-se uma duração média de 1/r. Uma vez que tenha tido acesso, a duração do acesso é assumido como sendo de 1 =L, em média. A figura 1.26, ilustra o processador interagindo com a memória comum, através do modelos SPN [Bause 2002]. O funcionamento ocorre da seguinte forma:

    - P1 lugar representa o estado local do processador quando ele está executando privadamente;
    - P2 lugar representa o estado local do processador quando ele está pronto para iniciar o acesso à memória comum;
    - P3 lugar representa o estado quando o processo está usando a memória comum;
    - O lugar P4 representa o estado local da memória comum quando não está em uso;
    - A transição T1 representa a ação do processador executando privadamente, a taxa de transição é l;
    - transição T2 representa o processador de ganhar acesso à memória comum; a taxa de esta transição é r;
    - A transição T3 representa o processador acessar a memória comum, a taxa de transição é m.

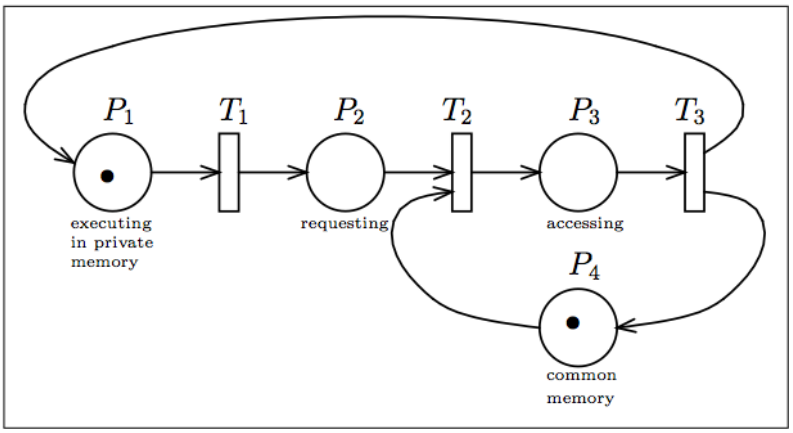


Figura 6.26. rede de Petri Estocástica representando um único processador em um sistema de memória compartilhada

2. A figura 6.27, ilustra um modelo SPN, onde a transição t1 está habilitada, como mostra o vetor de solições em M0(1,0,0,0,0). Tempo decorrido até o disparo de t1 é exponencialmente distribuido. No vetor M1 (0,1,1,0,0), t2 e t3 são ativadas ao mesmo tempo, por exemplo se t2 dispara mais rápido do que t3, as mudanças na rede fica sendo M2(0,0,1,1,0) e se t3 dispara antes de t2, obtemos a marcação M3(0,1,0,0,1) [Bause 2002] . A probabilidade desta ocorrecia é dada por:

P[t2 dispara em primeiro lugar M1] = P[2 < 3]

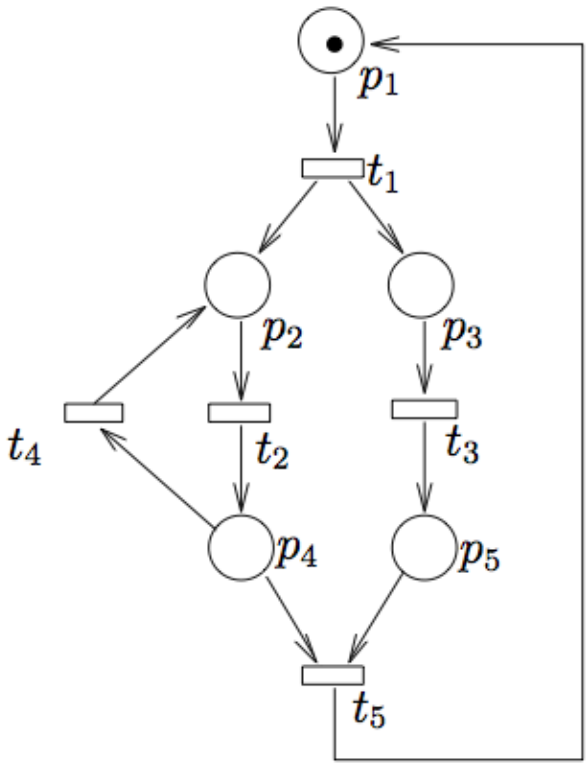


Figura 6.27. Exemplo Típico de SPN

6.6.1. Descrição do processo

6.6.2. Notas

A ferramenta apresentada neste capitulo foi TimeNET, mais existe vasta gama de outras ferramentas entre elas, estão o JARP [JARP 2001], CPNTools [CPNTool], o Integrated NetAnalyser (INA) [Starke 1999] , bem como as ferramentas exploradas neste trabalho, o TimeNET [Zimmermann and Freiheit 1998] [Hellfritsch 2009] , [Zimmermann and Knoke 2007], [Kelling 1995] e [Zimmermann et al. 2006].

6.6.3. Exercícios

1. Considere um sistema de manufatura. Existem seis unidades de processamento: processo de fabricação de embalagem (P1); processo de fabricação do corpo da caneta (P2); fabricação da ponta (P3); fabricação do tubo da carga (P4); fabricação da carga (P5) e processo de montagem (P6). Os processos P1 a P4 são realizados em paralelo; os processos P1 a P3 são interligados a P6. Por fim, P4 está ligado a P5 e consequentemente vai para P6. Represente o funcionamento do sistema de manufatura através de uma rede de Petri.

2.Em um determinado sistema de manufatura, quando a produção de uma peça é concluída o produto deve ser depositado em um estoque, caso tenha espaço disponível. A capacidade do estoque é de três peças. Depois de depositar a peça no estoque o produtor começa a produzir outra peça. Um consumidor, após consumir uma peça de cada vez, retira uma nova peça do estoque para ser consumido. Lembrando que após cada processo existe um buffer. Represente o funcionamento do sistema de manufatura através de uma RdP.

3. Modele o comportamento de um sistema computacional que possui dois processadores e utilizam uma memória compartilhada. O processador pode assumir os seguintes estados: não necessita de memória; solicita o uso da memória, mas não usa; ou usa a memória. Represente o funcionamento deste sistema através de uma rede de Petri.

4. Em um sistema de manufatura que possui duas empilhadeiras, tais empilhadeiras passam sobre duas máquinas. A máquina M1 só pode receber uma peça por vez e seu tempo de serviço TS é igual a duas unidades de tempo (UT). A máquina M2 pode receber duas peças por vez e seu tempo de serviço por peça é de 3unt. Represente o funcionamento deste sistema de manufatura através de uma rede de Petri.

5. Considere uma linha de produção. As peças chegam ao buffer 1, que em seguida são processadas na máquina A. Após processada, a peça é enviada para o buffer 2. Em seguida é enviada para a máquina 2 onde é realizado o segundo processamento. Represente o funcionamento deste sistema de manufatura através de uma RdP.

6. Considere um elevador de um edifício de 4 andares. Cada andar deve existir um sensor que detecta a presença do elevador e um botão de chamada do elevador. Os botões de controle no elevador são: 1,2,3 e 4 andar; parada de emergência e alarme. Quando o botão de emergência é acionado o elevador deve parar imediatamente e só volta a se movimentar quando for acionado um botão referente a um andar. O botão de alarme só funciona quando ocorre uma emergência. Represente o funcionamento deste sistema de manufatura através de uma RdP.

7. Em um determinado sistema de controle de bombas de água, o reservatório possui dois sensores. O sensor S1 avisa que o reservatório está cheio e sensor S2 informa que o reservatório está quase vazio. Existe um comando para acionar a bomba com o intuito de encher o reservatório. Além disso, há uma válvula de saída de água do reservatório. Essa válvula deve ser fechada quando o nível do reservatório estiver abaixo da sua capacidade indicada. Represente o funcionamento deste sistema de controle através de uma RdP.

8. Modele usando TimeNet o problema do barbeiro sonolento, assumindo que: (i) 5 cadeiras; (ii) o atendimento do barbeiro variando deterministicamente entre 3 a 6 minutos; e (iii) tempo de chega dos propriedades quantitativas/qualitativas e as interprete para o modelo.

6.7. Considerações Finais

Este capítulo apresentou uma introdução sobre redes de Petri, bem como definições, conceitos básicos e propriedades, as quais podem ser divididas em duas categorias:

propriedades comportamentais e propriedades estruturais. Em seguida, foram apresentadas as redes de Petri estocásticas (SPNs), que são de particular interesse deste trabalho.

As SPNs possuem transições com tempos exponencialmente distribuídos e transições imediatas. A ferramenta TimeNET foi apresentada e algumas instruções para criação e avaliação de modelos foram dadas. Para aumentar o conhecimento do leitor foram dados alguns exemplos práticos bem como sua resolução e para prática foram formulados alguns exercícios. Em Notas podem-se encontrar referências para demais conceitos. Por fim, alguns exercícios de fixação foram propostos com o objetivo de incentivar a prática fora da sala de aula.

6.8. Referências

[Ajmone Marsan et al. 1984] Ajmone Marsan, M., Conte, G., and Balbo, G. (1984). A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. ACM Trans. Comput. Syst., 2(2):93–122.

[Avizienis et al. 2004] Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. Dependable and Secure Computing, IEEE Transactions on, 1(1):11 – 33.

[Balbo 2001] Balbo, G. (2001). Introduction to stochastic petri nets.

[Bause 2002] F. Bause and P. S. Kritzinger. Stochastic Petri Nets - An Introduction to the Theory. 2002.

[Bertsekas and Tsitsiklis 2002] Bertsekas, D. P. and Tsitsiklis, J. N. (2002). Introduction to Probability.

[Brzezniak and Zastawniak 2009] Brzezniak, Z. and Zastawniak, T. (2009). Basic Stochastic Processes.

[CPNTool]CPNTools. *Computer Tool for Coloured Petri Nets.* http://wiki.daimi.au.dk/cpntools/. Acessado em: 02/11/2006.

[Desel and Reisig 1998] Desel, J. and Reisig, W. (1998). Place/Transition Petri Nets. Springer-Verlag.

[e K.Lautenbach 1978] e K.Lautenbach, H. (1978). Facts in place/transition nets. Proceedings of the Seventh Symposium on Mathematical Foundations of Computer Science.

[F.J.W.Simons 1978] F.J.W.Simons (1978). Modelling and analysis of communication protocols using numerical petri nets. Tese de Doutorado, Report 152, Department of Electrical and Engineering Science, University of Essex, Telecommunications System Group.

[German et al. 1995] German, R., Kelling, C., Zimmermann, A., and Hommel, G.(1995). Timenet: a toolkit for evaluating non-markovian stochastic petri nets. Performance Evaluation, 24:69 – 87.

[Goodman 1988] Goodman, R. (1988). Introduction to stochastic models. Benjamin/Cummings Pub- lishing Company, Inc,.

[Hellfritsch 2009] Hellfritsch, C. (2009). Timenet - examples of extended deterministic and stochastic petri nets.

[Higgins et al. 2002] Higgins, L., Mobley, K., and Smith, R. (2002). Maintenance Engineering Handbook.

[Howard] Howard, R. Dynamic probabilistic systems. Volume I: Markov Models. Volume II: Semi-Markov and Decision Processes. John Wiley Sons, Inc.,.

[JARP 2001] JARP: *Petri Nets Analizer*. Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, http://jarp.sourceforge.net. 2001.

[Jensen 1997] Jensen, K. (1997). A brief introduction to coloured petri net.

[Jensen et al. 2007] Jensen, K., Kristensen, L. M., and Wells, L. (2007). Coloured petri nets and cpn tools for modelling and validation of concurrent systems.

[Kelling 1995] Kelling, C. (1995). Timenet-sim-a parallel simulator for stochastic petri nets. In Simulation Symposium, 1995., Proceedings of the 28th Annual, pages 250–258.

[Kuo and Zuo 2003] Kuo, W. and Zuo, M. (New Jersey, 2003.). Optimal Reliability Modeling: Principles and Applications. ISBN 0-471-39761-X.

[Kurt and Jensen] Kurt and Jensen. Coloured petri nets and the invariant-method. Theoretical Computer Science, 14(3):317 – 336.

[Laprie et al. 1992] Laprie, J. C., Avizienis, A., and Kopetz, H., editors (1992). Dependability: Basic Concepts and Terminology. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Lilja 2000] Lilja, D. J. (2000). Measuring Computer Performance: A Practitioner's Guide. Cambridge University Press.

[Maciel et al. 1996] Maciel, P. R. M., Lins, R. D., and Cunha, P. R. F. (1996). Introdução às redes de petri e aplicações. X Escola de Computação, Campinas/SP Departamento de Informática, Universidade Federal de Pernambuco.

[Marsan et al. 1998] Marsan, M. A., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G. (1998). Modelling with generalized stochastic petri nets. SIGMETRICS Perform. Eval. Rev., 26:2–.

[Merlin and Farber. 1976] Merlin, P. M. and Farber., D. J. (1976). Recoverability of communication protocols - implications of a theorical study. IEEE Transactions on Communications, Cambridge, MA, USA.

[Molloy 1982] Molloy, M. K. (1982). Performance analysis using stochastic petri nets. IEEE Trans. Comput., 31:913–917.

[Murata 1989] Murata, T. (1989). Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77(4):541 –580.

[Noe and Nutt. 1976] Noe, J. D. and Nutt., G. J. (1976). Macro e-nets representation of parallel systems. IEEE Transactions on Computers.

[Peterson 1977] Peterson, J. L. (1977). Petri Nets, volume Vol. 9, No. 3,. Computing Surveys,.

[Peterson 1981] Peterson, J. L. (1981). Petri net theory and the modeling of systems. Englewood Cliffs, 26:2–.

[Petri 1966] Petri, C. (1966). Kommunication mit automaten.

[Rausand and Høyland 2004] Rausand, M. and Høyland, A. (2004.). System reliability theory: models, statistical methods, and applications. Wiley-IEEE.

[Rozenberg and Engelfriet 1998] Rozenberg, J. and Engelfriet, J. (1998). Elementary Net Systems, volume 1491 of p. 12-121. Lectures on Petri Nets I: Basic Models. Advances in Petri Nets, Lecture Notes in Computer Science, Springer-Verlag.

[Starke 1999] STARKE, P. H., ROCH, S. *INA - Integrated Net Analyzer*. Humbolt Universitat zu Berlin, Institut für Informatik. 1999.

[Trivedi et al. 2009] Trivedi, K., Kim, D. S., Roy, A., and Medhi, D. (2009). Dependability and security models. In Design of Reliable Communication Networks, 2009. DRCN 2009. 7th International Workshop on, pages 11 –20.

[World 2010] World (2010). Petri nets world. Maintained by TGI group at the University of Hamburg.

[Zimmermann and Freiheit 1998] Zimmermann, A. and Freiheit, J. (1998). Timenetmsan integrated modeling and performance evaluation tool for manufacturing systems. In Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on, volume 1, pages 535 – 540 vol.1.

[Zimmermann and Knoke 2007] Zimmermann, A. and Knoke, M. (2007). User manual timenet 4.0.

[Zimmermann et al. 2006] Zimmermann, A., Knoke, M., Huck, A., and Hommel, G. (2006). Towards version 4.0 of timenet. Measuring, Modelling and Evaluation of

Computer and Communication Systems (MMB), 2006 13th GI-ITG Conference, pages 1–4.

[Zuberek 1980] Zuberek, W. M. (1980). Timed petri nets and preliminary performance evaluation. In Proceedings of the 7th annual symposium on Computer Architecture, ISCA ’80, pages 88–96, New York, NY, USA. ACM.

Capítulo 7

Introdução as Redes de Sensores Sem Fio

Matheus L. Araújo, Marianna A. Araújo, Dayanne K. F. R. Escalé

Resumo

O estudo das Redes de Sensores Sem Fio (RSSF) tem atraído atenção de pesquisadores e da indústria devido à variedade de aplicações suportadas por esta tecnologia, que pode abranger as seguintes áreas: militar, médica, extração de petróleo e gás, meio ambiente, entre outras. Este crescente interesse está associado ao surgimento e consolidação da computação ubíqua e pervasiva, bem como, às vantagens no uso das RSSF, como por exemplo, a eliminação de altos custos com cabeamento e a facilidade de implantação em ambientes hostis.

Visto a importância das RSSF, este minicurso objetiva fundamentar teoricamente seus participantes, apresentando: conceitos básicos relacionados a estas redes; protocolos de comunicação utilizados; tecnologias que permitem simular e desenvolver aplicações para RSSF; os desafios de projetos para RSSF com ênfase no tratamento de tolerância à falha.

Para demonstrar o comportamento de uma RSSF, será executada uma aplicação desenvolvida na linguagem de programação NesC. A execução será feita no ambiente do sistema operacional TinyOS, simulado pelo TOSSIM. O TOSSIM é um simulador para RSSF amplamente utilizado e desenvolvido pela equipe de desenvolvimento do TinyOS, para simulação de aplicações na falta de plataformas de hardware dos nós.

7.1. Objetivos do Curso

O curso tem como objetivos apresentar aos participantes o conceito de Redes de Sensores Sem Fio, bem como despertar o interesse dos mesmos pela área, de maneira que novos pesquisadores sejam formados para cooperar na pesquisa deste campo de pesquisa.

7.2. Tipo de Curso

Teórico

7.3. Tratamento dado ao tema

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

Apanhado geral do tema, introduzindo o mesmo de uma maneira lenta e aprofundando envolvendo os participantes no conteúdo aos poucos,

7.4. Perfil dos participantes

Alunos de Graduação Básica em Cursos de Computação

7.5. Infraestrutura física necessária para a apresentação

Notebook (do apresentador) e projetor multimídia.

7.6. Introdução

Os avanços nas áreas de microprocessadores, sistemas eletromecânicos (*MEMS – Micro Electro-Mecanical Systems*) e comunicação sem fio, além dos novos materiais de sensoriamento impulsionaram o desenvolvimento das Redes de Sensores Sem Fio (RSSF), bem como o seu uso em diversas áreas [LOUREIRO et al., 2003]. Ambientes como o da extração de petróleo, do controle da biodiversidade, e do controle de ambientes industriais, ou mesmo ambientes hostis e que tragam riscos à vida humana podem ser monitoradas com o uso das RSSF. Nestes ambientes, sensores estão sendo utilizados para coleta remota de dados. Muitas vezes os locais de onde serão coletados os dados são de difícil acesso e/ou oferecem riscos para o rompimento dos cabos de sensores comuns. Desta forma torna-se mais viável a utilização das RSSF nestes ambientes. Atualmente estas redes são utilizadas em muitas aplicações industriais e de consumo, tais como monitoramento de processos industriais e de controle, vigilância do status de funcionamento de máquinas, dentre outros.

As RSSF podem ser definidas como um agrupamento de vários pequenos equipamentos com capacidade de sensoriamento, poder computacional e dotados de comunicação sem fio. Estes pequenos equipamentos podem ser denominados nós [AKYILDIZ et al., 2002]. Além disso, essas redes ainda possuem a capacidade de auto-organização em redes *ad hoc*. Já estas são redes sem infraestrutura nas quais os elementos trocam informações diretamente sem que exista um ponto de acesso centralizado [VELLOSO et al., 2003]. São vários os fatores que diferenciam as Redes de Sensores Sem Fio das redes comuns. Dentre eles podem ser destacados o alto número de nós sensores distribuídos, a restrição no poder do hardware, a característica de possuir mecanismos de autoconfiguração e fatores críticos como a limitação da capacidade energética e a tolerância a falhas.

Nas RSSF, os nós possuem baixo poder de processamento, por isso necessitam de algoritmos otimizados para compensar suas limitações de hardware. Cada um dos nós é equipado com uma variedade de sensores tal qual seja necessário à aplicação. Os sensores podem ser utilizados para medir diferentes tipos de fenômenos físicos ou químicos, como por exemplo, o acústico, o sísmico, a temperatura, a pressão, entre outros.

Os nós, de maneira isolada possuem pouca capacidade computacional e de energia, mas um esforço colaborativo entre eles permite a realização de tarefas maiores que não seriam realizadas pelos mesmos individualmente [RUIZ et al., 2004]. De acordo com a necessidade da aplicação os nós sensores podem ser lançados e estar dispostos em ambientes inóspitos ou de difícil acesso, onde os mesmos se organizam automaticamente formando uma rede *ad hoc* para a coleta de dados sobre o fenômeno desejado. Esses nós podem realizar um processamento local sobre os dados coletados e por fim realizam o

envio das informações ao observador através do ponto de acesso mais próximo, como pode ser visto no exemplo da Figura 7.1.

Um ponto de acesso é um nó que na maioria dos casos difere dos outros nós sensores por ter a finalidade de coletar as informações da rede e enviar para um observador. Ele conecta a RSSF com o mundo exterior exercendo o papel de gateway. Em geral os nós que funcionam como pontos de acesso possuem uma maior capacidade energética, e conseguem atingir um maior alcance de comunicação, porém não possuem funcionalidade de sensoriamento [VIEIRA et al., 2003]. Os pontos de acesso também são conhecidos como sorvedouro ou sink.

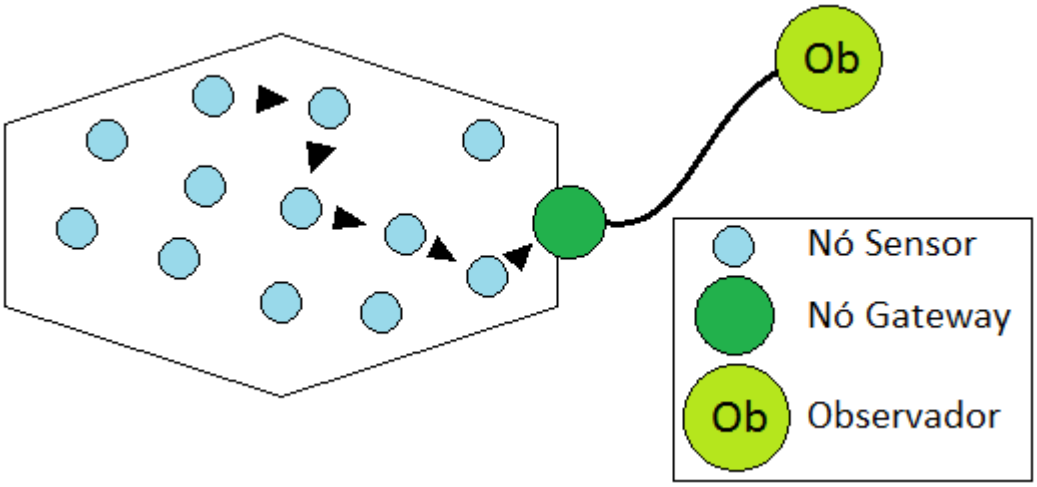


Figura 7.1 Rede de Sensores Sem Fios

De acordo com [SOARES, 2007] o maior desafio para as RSSFs está relacionado com a limitação de recursos nos nós, que é uma consequência das dimensões reduzidas dos dispositivos sensores. Dentre estas limitações, a energia principalmente deve ser administrada de maneira eficiente, para que se tenha uma longa autonomia e durabilidade da rede.

Além das limitações de recursos, outros fatores que podem afetar a autonomia e durabilidade de uma RSSF estão relacionados com falhas que a rede pode apresentar. Em [MACEDO et al., 2005] são relacionadas possíveis falhas que podem ocorrer nestas redes, tais como, fontes móveis de interferência que podem afetar a comunicação da rede, quebra acidental dos nós, desastres naturais que podem ocasionar a perda de nós, falha no hardware dos dispositivos deixando-o inoperante, falhas maliciosas advindas de ataques, e esgotamento de energia, estas falhas serão melhor explicadas nos próximos parágrafos.

Conciliar técnicas de tolerância a falhas e ao mesmo tempo respeitar as limitações dos dispositivos da rede, se torna um desafio. A tolerância a falhas é um fator essencial para a transmissão dos dados de maneira satisfatória nestas redes, visto que o ambiente no qual elas venham a estar inseridas pode ser propício a defeitos.

A maneira como o roteamento é realizado é essencial para o bom funcionamento da rede e também para evitar falhas na mesma, pois ele é quem definirá o caminho a ser tomado pelos dados coletados até que estes atinjam o sorvedouro, além de ser ele o responsável por tomar medidas para que durante este trajeto não ocorra perda de informações. Medidas essas que podem ser, por exemplo, a duplicidade no envio dos dados sendo esta caracterizada pelo envio para vários nós em simultâneo, evitando assim que caso um nó sofra dano a informação deixe de atingir o destino. Os protocolos de roteamento servem para organizar e gerenciar o tráfego de informações na rede roteando os pacotes.

7.7. Redes de Sensores Sem Fio

As RSSF são redes formadas por nós sensores que agem de forma colaborativa para monitorar o ambiente ao qual seus dispositivos estejam dispostos. Atuar de forma colaborativa significa que os sensores de forma singular não possuem muito poder, mas quando atuam em conjunto, trocando informações entre si, podem ter grande eficiência no contexto da aplicação [NAKAMURA, 2003]. Os nós sensores são dispositivos que se caracterizam por possuir as capacidades de sensoriamento (sensores e atuadores), armazenamento, processamento, comunicação, e fonte de energia própria [DULMAN et al., 2006; BHATTACHARYA et al., 2003].

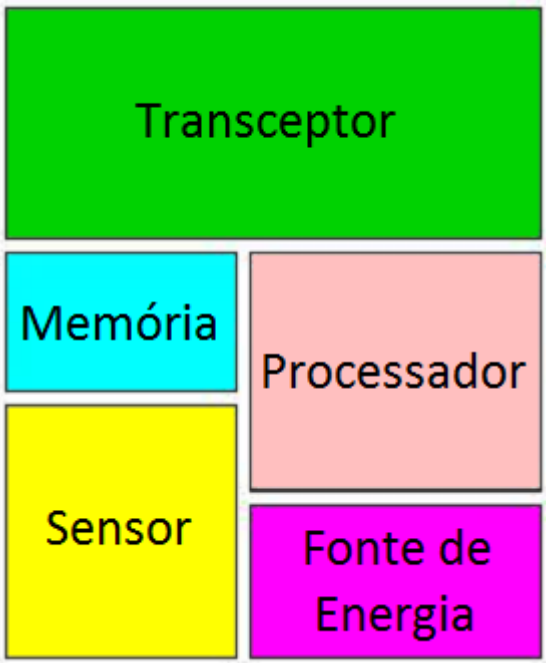


Figura 7.2 Hardware Básico de um Nó Sensor

A Figura 7.2 mostra os componentes básicos de um nó sensor descritos em [LOUREIRO et al., 2003; VIEIRA, 2004], sendo eles o transceptor, responsável por realizar a comunicação sem fio. O microprocessador, que responde pela capacidade de processamento no nó. Quanto à memória, o microcontrolador possui uma interna e, na maior parte das vezes uma memória externa e secundária também está presente no nó. Os sensores são os responsáveis por realizar o mapeamento (registro) de eventos do mundo real em dados que serão coletados e utilizados pela rede. A fonte de alimentação dos componentes do hardware normalmente é composta por uma bateria (recarregável ou não) que pode chegar a ter o tamanho de uma moeda. Esta fonte de alimentação é um dos itens mais importantes no contexto das redes de sensores, pois a durabilidade da RSSF depende da energia disponível.

De acordo com [BERNDT, 2008; FAÇANHA, 2007], os nós sensores normalmente possuem baixa frequência de amostragem, ou seja, a quantidade de dados por amostra/coleta é pequena em cada envio. Em cada nó, também está presente um conversor analógico/digital que faz a conversão dos dados colhidos pelos sensores para um formato digital para que estes possam ser tratados pelo microcontrolador. Este se diferencia de um microprocessador por não somente realizar operações lógica/aritméticas, mas também por realizar o controle do funcionamento do nó sensor como um todo.

Os nós, além do hardware com capacidade limitada, também possuem software, como sistema operacional, protocolos de comunicação, e uma aplicação desenvolvida para uma situação especifica.

O fato dos nós possuírem limitações em seu hardware faz com que este tipo de rede seja diferente dos modelos de redes tradicionais, possuindo, desta forma, características próprias que merecem ser abordadas na próxima seção.

7.8. Características das RSSF

Em Redes de Sensores Sem Fio existem características inerentes a cada área em que as mesmas são aplicadas, desta forma ocorrem questões particulares para cada aplicação. Entender as características da RSSF é importante, pois essa classificação será utilizada no decorrer deste trabalho. Em [RUIZ et al, 2004; SANTOS, 2008] são apresentadas essas características, sendo as mesmas descritas a seguir:

7.9. Endereçamento dos sensores

Dependendo da aplicação, pode existir a necessidade de se endereçar cada sensor unicamente ou não. Por exemplo, sensores para monitorar a temperatura de determinados ambientes devem ser endereçados unicamente, pois é necessário saber de onde os dados estão sendo coletados. Em outra situação, sensores utilizados para detectar movimento em um ambiente fechado não precisam ter identificação única, pois o interesse da aplicação é apenas saber se no ambiente está ocorrendo algum movimento e não o local exato de onde o mesmo foi detectado.

7.10. Agregação de dados

É a capacidade que uma RSSF possui de reunir ou sumarizar os dados que serão coletados pelos sensores. Se a rede possuir essa funcionalidade, o número de mensagens que precisam ser transmitidas por ela diminui e junto a isto também diminui o consumo de energia. Os dados coletados são agregados e sumarizados ainda na rede, antes do envio para a estação base. Esta situação pode ser vista na Figura 7.3.

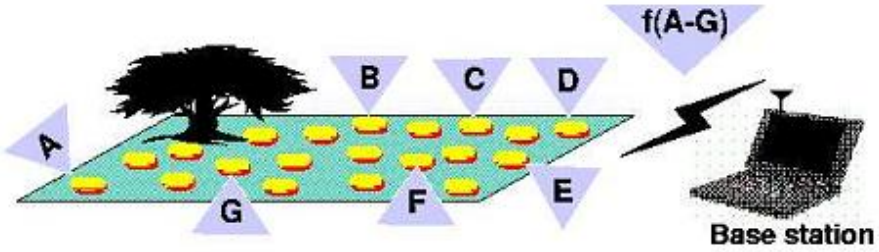


Figura 7.3 Agregação dos dados coletados pelos sensores

7.11. Mobilidade dos sensores

Refere-se à possibilidade de os sensores da rede serem moveis ou imóveis com relação ao sistema onde os dados serão coletados. Por exemplo, sensores colocados flutuando sobre o oceano para monitorar o nível de poluição da água são móveis, enquanto que sensores colocados em uma sala de servidor para medir a temperatura da refrigeração são estáticos. Todos os nós presentes na rede, mesmo que sejam imóveis possuem a capacidade de se reconfigurar na rede, pois caso algum dos nós falhe, a sua falta não comprometerá a rede.

7.12. Restrições dos dados coletados

Os dados coletados podem ter algum tipo de restrição, como por exemplo, um intervalo de tempo máximo para que os valores coletados cheguem até uma entidade de supervisão, isso pode ocorrer no caso de um ambiente crítico em que a temperatura varia constantemente e que ao atingir certo nível a entidade deve saber imediatamente, em no máximo alguns segundos.

7.13. Quantidade de sensores

A quantidade de sensores que serão utilizados depende da aplicação, se for o caso do monitoramento de ambientes tais como florestas e oceanos, por exemplo, pode-se estimar a quantia de sensores entre 10 e 100 mil unidades. Neste ponto nota-se a importância da escalabilidade para as Redes de Sensores Sem Fio, embora numa grande porcentagem das aplicações os sensores sejam estáticos em relação ao sistema de sensoriamento.

7.14. Tipo de comunicação Broadcast

As informações são enviadas pelos dispositivos de sensoriamento para os seus vizinhos que retransmitem as mesmas até que estas cheguem ao destino. Mesmo as informações passando por outros nós, esta somente é tratada ao chegar ao destino.

7.15. Limitação de recursos

As RSSF possuem a característica da baixa oferta de recursos, pois devido ao tamanho dos dispositivos de sensoriamento, as capacidades de processamento, armazenamento e energia dos nós é limitada. Dentre os itens citados, a questão da energia é a que mais sofre com a limitação de recursos, pois em uma grande quantidade de aplicações, os sensores utilizados serão dispostos em áreas de difícil acesso, o que impossibilita a manutenção dos nós, desta forma tornando a energia um fator crucial para a rede. As aplicações, programações de sensores, e protocolos utilizados devem ser escolhidos levando-se em consideração o seu consumo energético.

7.16. Dependência da aplicação

De acordo com o objetivo para o qual a rede é proposta os parâmetros de configuração, operação e manutenção variam. Todo o projeto de hardware e software da rede deve ser planejado de acordo com o proposito da mesma, a fim de se atingir o menor custo de produção e se ter uma rede eficiente.

7.17. Classificações das RSSF

Segundo [RUIZ, 2004], as RSSF podem ser classificadas de acordo com alguns critérios como mostrado na Tabela 7.1. Nesta tabela estão relacionados os critérios utilizados para a divisão em categorias, a classificação atribuída a cada rede pelo critério selecionado e a descrição de cada tipo.

Tabela 7.1 Classificação das RSSF

Critério	Classificação	Descrição

Composição	Heterogênea	Os nós apresentam diferentes capacidades de hardware.
	Homogênea	Os nós podem ser idênticos quanto ao hardware, e por vezes podem apresentar softwares diferentes.
Organização	Plana	Os nós da rede não formam grupos neste tipo de organização.
	Hierárquica	A organização é feita agrupando os nós em *clusters*. Em cada *cluster* existirá um *cluster-head* que pode ser eleito pelos demais nós. Os *clusters* podem se organizar em hierarquias.
Distribuição	Regular	Os nós da rede são distribuídos de maneira uniforme na área monitorada.
	Irregular	Os nós são distribuídos de maneira não uniforme na área monitorada.
Mobilidade	Estática	Os elementos depois de depositados permanecem imóveis durante o período de vida da rede.
	Dinâmica	Os nós podem se deslocar depois de depositados para o monitoramento (apresentam mobilidade).
Densidade	Balanceada	A quantidade de nós presentes por área é semelhante em toda a área de monitoramento.
	Densa	A rede apresenta uma alta quantidade de nós por área, sendo esta concentrada.
	Esparsa	A rede apresenta uma baixa quantidade de nós por área.
Controle	Aberta	A rede realiza apenas o monitoramento na região.
	Fechada	A rede realiza o monitoramento e também atua na região.
Coleta de dados	Periódica	Em intervalos de tempo regulares são coletados dados na rede.
	Contínua	A coleta de dados é feita continuamente, sem pausas.
	Reativa	A coleta de dados é realizada apenas quando um evento ocorre ou ainda quando o observador solicita.
	Tempo-real	Tem como objetivo a coleta da maior quantidade de dados possível dentro do menor intervalo de tempo.
Disseminação	Programada	A disseminação dos dados ocorre em intervalos de tempo programados.
	Contínua	Os dados são disseminados constantemente.
	Sob-demanda	Os dados são disseminados quando ocorrem eventos ou quando o observador realiza consulta.

7.18. Pilha de Protocolos das RSSF

A pilha de protocolos da RSSF é abordada no trabalho de alguns autores como [AKYILDIZ et al., 2002], [DELICATO, 2005], [SANTOS, 2008], [JABOUR, 2009] e [NETO, 2010]. Esta pilha é utilizada nos nós sensores, bem como no sorvedouro e é formada conforme a Figura 7.4, contendo as camadas de aplicação, transporte, rede, enlace e física, além dos planos de gerenciamento de energia, mobilidade e tarefas.

Os planos de gerenciamento existem para permitir que os sensores trabalhem de maneira eficiente, evitando o consumo excessivo de energia, gerenciando a movimentação dos sensores dentro da rede e compartilhando os recursos na rede.

Na pilha de protocolos existem os planos de gerenciamento, sendo eles responsáveis pela energia, mobilidade e tarefas. O plano de gerenciamento de energia é responsável por regular o consumo da mesma, sabendo que a energia é um dos fatores primordiais nas RSSF esse plano é essencial ao seu funcionamento, por meio deste plano um sensor pode desligar o transceptor após o recebimento de uma mensagem, no intuito de economizar energia, e ainda caso a energia do nó esteja baixa, este pode enviar um sinal em *Broadcast* para os demais nós para que este não seja mais utilizado para encaminhar informações ao nó sink advindas de outros nós. A gerência de mobilidade permite aos nós que estes saibam informações e rotas das redes, além de identificar os nós vizinhos. Como os sensores podem ser móveis na rede, esta pode sofrer alterações constantes na disposição dos nós, assim é essencial que cada nó conheça a disposição dos vizinhos para saber o melhor caminho para transmitir a mensagem ao sink. E o plano de gerencia de tarefas é responsável pelo escalonamento de tarefas entre os sensores da rede, uma vez que dependendo do alcance dos dispositivos de sensoriamento e da proximidade dos nós fica desnecessário o monitoramento da mesma área por vários nós, além de que se muito próximos, esse monitoramento irá gerar redundância de informações para a rede e ainda um consumo maior de energia.

Figura 7.4 Pilha de Protocolos em RSSF

A camada física é responsável por converter os bits que compõem a informação vinda do sensoriamento em sinais mais adequados para a transmissão através do ar (sem fio). Esta camada lida com a seleção da frequência, geração de portadora, detecção e modulação de sinal, e criptografia de dados. Desta forma, a camada física agrega técnicas simples, mas robustas de transmissão, recepção e modulação. É uma camada importante, pois o consumo de energia é maior durante a transmissão e recepção dos dados, assim as técnicas utilizadas nesta devem ser otimizadas e pensadas para a economia de energia sem comprometer a transmissão dos dados.

A camada de enlace é responsável pelas atividades de controle de acesso ao meio de comunicação, multiplexação/demultiplexação de dados, detecção de quadros e controle de erros. Como protocolo de controle de acesso ao meio pode-se citar o MAC (*Medium Access Control*) que atua nesta camada. Causas de desperdício de energia nesta camada são: colisões, recebimento de quadros endereçados a outros destinatários e espera ociosa segundo (SANTOS, 2008).

A camada de rede trata do roteamento dos dados requisitados pela camada de transporte. Muitos protocolos de redes têm sido propostos e novos sempre estão surgindo devido à importância desta camada para redução do consumo energético. Protocolos presentes nesta camada são o LEACH, LEACH-C, TEEN e ICA, entre outros, estes são protocolos de roteamento hierárquico. Também são presentes nesta camada protocolos de roteamento plano como o EAR, PEGASIS, PROC, SPIN, entre outros. E ainda podemos encontrar nesta camada protocolos híbridos como SID e o EF-Tree.

A camada de transporte é responsável por ajudar a manter o fluxo de dados, quando requerido pela aplicação, ou seja, proporciona a entrega de dados confiável fim-a-fim (entre a origem e o sorvedouro).

Por fim, a camada de aplicação tem a funcionalidade de abstrair a topologia física da RSSF para as aplicações, além de prover vários serviços para estas, como serviço de localização, agregação de dados, desligamento de sensores, sincronização do tempo e configuração da rede.

7.19. Áreas de Aplicação de Redes de Sensores Sem Fio

As aplicações para Redes de Sensores Sem Fio são variadas e podem ser desenvolvidas utilizando um ou mais tipos de nós sensores. Com relação aos nós sensores, os mesmos podem ser homogêneos ou heterogêneos em relação às dimensões, tipos, funcionalidades e quantidade de sensores, com relação às dimensões elas também podem variar de acordo com o estado atual da tecnologia de fabricação utilizada (FAÇANHA, 2007). A seguir serão descritas algumas áreas de aplicação para as RSSF dadas em (AKYILDIZ et al., 2002; LOUREIRO et al., 2003; FAÇANHA, 2007):

7.19.1. Militar

A rápida implantação, a auto-organização, e a tolerância a falhas característica nas redes de sensores, fazem delas uma técnica de sensoriamento promissora para o comando, controle, comunicação, computação, inteligência, vigilância, reconhecimento e sistemas de identificação militares. Neste tipo de aplicação, os requisitos de segurança são fundamentais. O alcance das transmissões dos sensores é limitado propositalmente para evitar escutas não autorizadas. Os dados são criptografados e submetidos a processos de assinatura digital. As dimensões são extremamente reduzidas e podem utilizar nós sensores móveis como os transportados por robôs.

7.19.2. Medicina/Biologia

Nós sensores podem ser implantados para monitorar pacientes em tratamento e ajudar outros pacientes com algum tipo de deficiência. Também podem ser utilizados, por exemplo, no monitoramento do funcionamento de órgãos como o coração (ver Figura 7.5) e detectar a presença de substancias que indicam a presença ou surgimento de um problema biológico, seja no corpo humano ou animal.

7.19.3. Controle Industrial

Podem-se utilizar as RSSFs para realizar algum mecanismo de controle, este pode ser aplicado, por exemplo, em ambientes industriais. Uma aplicação seria a utilização de sensores sem fio embutidos em "peças" numa linha de montagem para fazer testes no processo de manufatura, verificando a temperatura no interior das máquinas.

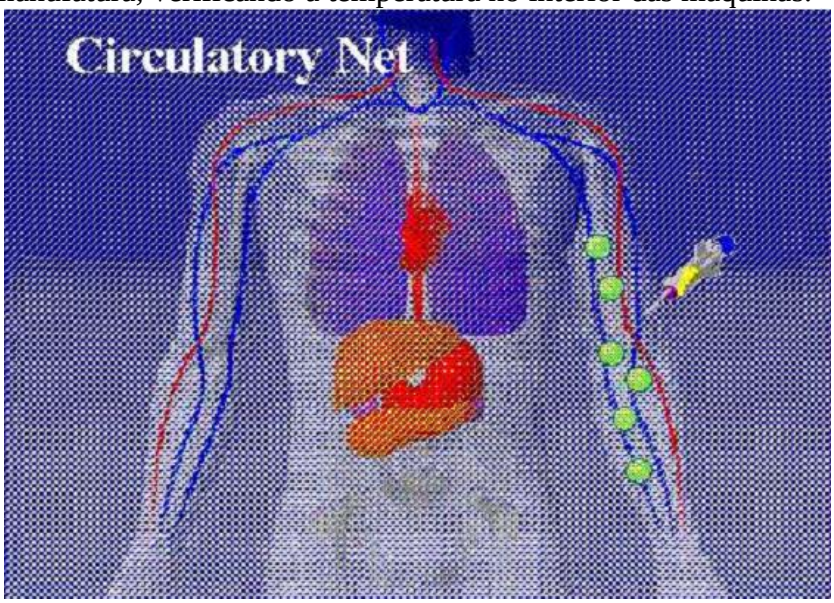


Figura 7.5 Sensores Introduzidos no Corpo Humano para Monitorar Condições Físicas

7.19.4. Ambiente

Para monitorar variáveis ambientais em locais internos como prédios e residências, e locais externos como florestas, desertos, oceanos, vulcões, dentre outros. Em ambientes externos, uma aplicação seria o monitoramento de um vulcão que quando as condições indicassem uma erupção, a população seria notificada para evacuar a área salvando vidas desta maneira.

7.19.5. Tráfego

Para monitorar tráfego de veículos em rodovias, malhas em vias urbanas. Para melhorar o fluxo de transito em determinado local, para evitar congestionamentos. No exemplo da Figura 7.6 vemos uma aplicação que monitora a quantia de veículos que entra na estrada e a que sai também (levando em consideração que a bifurcação leva ao mesmo destino no

final), e de acordo com as informações o painel com a seta vermelha exibe qual a via com menor quantidade de automóveis, desta forma evitando engarrafamentos.

7.19.6. Segurança

Para prover segurança em centros comerciais, estacionamentos por exemplo. Nestes ambientes as redes podem exercer o papel de monitorar a presença de pessoas.

Também podemos citar como exemplos de aplicações para as RSSF o monitoramento de ambientes refrigerados onde a temperatura deve ser constante, a gerência de inventários, o uso em brinquedos interativos que fazem uso de sensoriamento, o monitoramento da qualidade de produtos, entre outros.

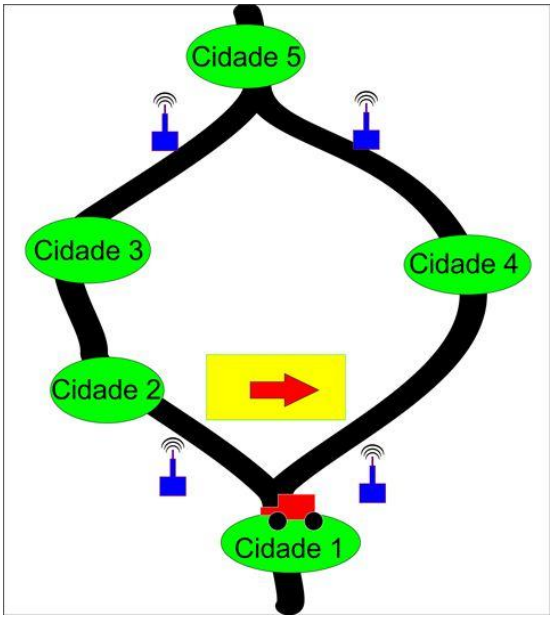


Figura 7.6 RSSF sendo utilizada para o controle de tráfego em pedágios

7.20. Exemplos de Setores de Aplicação de RSSF Já Existentes

Como citado em (LOUREIRO et al., 2003) várias áreas já aplicam Redes de Sensores Sem Fio, como as listadas a seguir:

7.20.1. Produção industrial

Em indústrias petroquímicas, fábricas, refinarias e siderurgias em geral pode ser feito o monitoramento de parâmetros como fluxo, pressão, temperatura, e nível, podendo identificar problemas como vazamento e/ou aquecimento (Veja a figura 7.7).

Figura 7.7 Produção Industrial

7.20.2. Áreas industriais

Redes de Sensores podem ser adotas para o monitoramento de dados em áreas que sejam de difícil de acesso ou que ofereçam riscos ao ser humano (Veja a figura 7.8).

Figura 7.8 Zonas Industriais

7.20.3. Extração de petróleo e gás

Na indústria de petróleo e gás, centenas de variáveis físicas são monitoradas, sendo esse monitoramento importante para o funcionamento da indústria. O mecanismo de sensoriamento utilizado na monitoração dessas indústrias é cabeado e por ocupar um grande espaço físico, as RSSF estão sendo bastante exploradas nesta área no intuito de reduzir o espaço ocupado (SILVA, 2006) (Veja a figura 7.9).

Figura 7.9 Extração de Petróleo e Gás Natural

7.20.4. Indústria de aviação.

Na indústria de aviação, é crescente o uso da tecnologia denominada *fly-by-wire*, em que transdutores (sensores e atuadores) são muito utilizados. O problema encontrado está na grande quantidade de cabos utilizados para a interconexão. Para solucionar o problema, sensores sem fio estão começando a serem adotados. Situação exemplificada na figura 7.10.

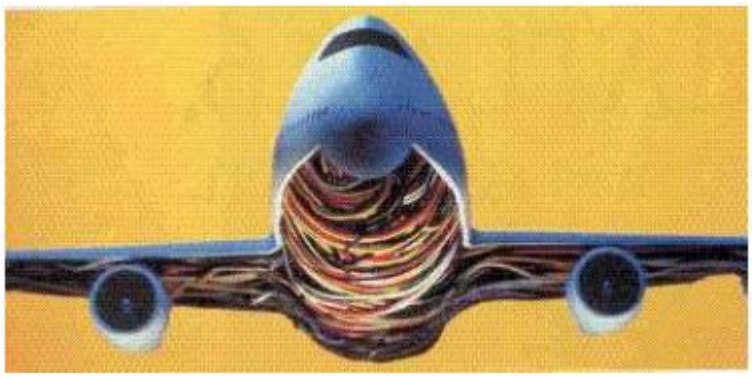


Figura 7.10 Industria de Aviação

Redes de Sensores Sem Fio tendem a executar tarefas colaborativas, onde o conjunto das informações coletadas irá gerar o resultado desejado. Em sua grande maioria a aplicação é quem define os objetivos das RSSFs, porém podem-se destacar algumas atividades que são normalmente executadas por essas redes.

7.20.5. Sistema Operacional TinyOS

Em [RUIZ et al., 2004] uma descrição sobre o TinyOS é feita, sendo esta seção baseada no conteúdo do artigo citado. O sistema operacional TinyOS é simples, compacto baseado em eventos e foi projetado para ser utilizado essencialmente em Redes de Sensores Sem Fio, atendendo às suas necessidades como operações intensivas de concorrência com baixo poder de hardware (que possui limitações de recursos como memória de programa semelhante a 8KB e/ou memória RAM na faixa de 512 bytes) e economia de energia. A linguagem de programação adotada pelo TinyOS é a nesC (Pronuncia-se “NES-see”).

7.20.5.1. Histórico do TinyOS

O TinyOS teve seu desenvolvimento iniciado por Jason Hill, em seu projeto de Mestrado na UC Berkeley (*University of California, Berkeley*) no ano de 1999. E sofreu várias atualizações, na Tabela 7.2 pode-se observar as atualizações lançadas do TinyOS em ordem cronológica.

Tabela 7.2 Histórico do TinyOS

Data	Acontecimento
1999	Primeira plataforma TinyOS (WeC) e implementações do Sistema Operacional são desenvolvidos em Berkeley.
2000	Berkeley lança a plataforma rene e parceiros como a Crossbow Inc., passam a produzir hardware. É disponibilizada ao público através do SourceForge a versão 0.43 do TinyOS. As versões anteriores à 1.0 do TinyOS são uma mistura de scripts em C e Perl.
2001	Berkeley desenvolve a plataforma mica e libera ao público a versão 0.6 do TinyOS.
Fev.2002	Berkeley distribui 1000 nós mica para outros participantes do projeto NEST (*Network Embedded Systems Technology*).
Abr.2002	Um trabalho sobre a linguagem de programação nesC é iniciado em colaboração entre a Intel Research e a UC Berkeley.
Set.2002	É lançado a versão 1.0 do TinyOS implementado em nesC.
Ago.2003	A versão 1.1 do TinyOS é lançada, tendo novos recursos em nesC, incluindo dados de detecção de corrida.
Set. 2003 – Dez.2005	O TinyOS adota um processo de liberação periódica menor.
Jun.2004	Os grupos de trabalho para os próximos passos do TinyOS são formados baseados em experiência de porta-lo para novas plataformas. Iniciam-se os trabalhos para a versão 2.0.
Jul.2005	Concluído o projeto NEST.
Dez.2005	Lançada a versão 1.1.15 do TinyOS, sendo esta a ultima versão 1.1.
Fev.2006	É lançado a versão beta1 do TinyOS 2.0 no 3º TinyOS Technology Exchange em Stanford, CA.
Jul.2006	Lançado o beta2 do TinyOS 2.0.
Nov.2006	O TinyOS 2.0 é lançado na Conferência SENSYS em Boulder, CO.
Abr.2007	Durante o 4º TinyOS Technology Exchange em Cambridge, MA, a versão 2.0.1 do TinyOS é lançada.
Jul.2007	Lançado o TinyOS 2.0.2. Iniciados os trabalhos para o TinyOS 2.1.
Ago.2008	TinyOS 2.1.0 é lançado.
Abr.2010	É lançada a versão 2.1.1 do TinyOS.
Jul.2010	O TinyOS é movido de servidor, passando a ser hospedado no Google Code, parte da transição incluiu a colocação de todas as partes do TinyOS sob uma nova licença BSD (no SourceForge várias licenças compatíveis foram utilizadas).

7.20.5.2. O que é TinyOS

O TinyOS é um sistema operacional simples, gratuito e de código aberto, baseado em componentes e feito para trabalhar com Redes de Sensores Sem Fio. O sistema operacional faz uso de eventos e de um conjunto de serviços, como descrito a seguir.

O TinyOS possui um escalonador de tarefas baseado em uma fila (FIFO – "*First In First Out"*), que utiliza uma estrutura de dados com tamanho limitado. Este escalonador é configurado para escalonar tarefas para execução sempre que o processador ficar disponível. Também não há preempção entre as tarefas, de maneira que uma vez executada a tarefa deve continuar em execução até ser concluída (*Run to Completion*). Desta forma as tarefas devem ser curtas para não ficar em execução por tempo indeterminado. Para garantir o termino das tarefas, esta também não pode bloquear recursos, além de não poderem entrar em espera ocupada.

Os eventos no TinyOS, da mesma forma que as tarefas, também são executados até o seu termino, porem estes podem realizar a preempção de tarefas ou outros eventos. Um evento pode sinalizar que um serviço foi concluído ou ainda a ocorrência de um evento no ambiente onde a rede está instalada. Os componentes de baixo nível são conectados diretamente as interrupções de hardware através de tratadores, onde as interrupções podem ser externas (eventos do temporizador) ou eventos contadores. São tarefas que o tratador pode executar: depositar informações no ambiente, escalonar tarefas, sinalizar outros eventos ou chamar comandos. Sempre que ocorre uma interrupção de hardware um tratador é executado e continua em execução até ser concluído, podendo interromper tarefas e ainda outros tratadores de interrupção. Pela razão de as tarefas no TinyOS não serem preemptivas, não existindo operações de bloqueio todas as operações de longa duração devem ser divididas em etapas (fases).

O TinyOS também é um conjunto de serviços. Os principais serviços implementados por interfaces e componentes estão disponíveis junto ao TinyOS, sendo eles:

- Rádio, MAC, Mensagens, Roteamento: Componentes que implementam a pilha de protocolos do TinyOS.
- Interface de Sensores: Interfaces para os vários tipos de sensores que podem ser usados nos nós sensores com TinyOS.
- Gerência de Energia: A energia é um dos recursos mais importantes para as RSSF.
- Depuração: Componentes e ferramentas disponibilizadas para deputação de código.
- Temporizadores: Componentes disponibilizados que fornecem acesso aos relógios do nó sensor.

Para se aprofundar mais sobre o TinyOS uma excelente fonte de informação é o "*TinyOS Documentation Wiki*" (*http://docs.tinyos.net/index.php/Main_Page*), além do próprio site oficial do TinyOS (*http://www.tinyos.net*).

7.21. A Linguagem de Programação nesC

Programar para TinyOS pode ser um pouco trabalhoso no início visto que é necessário aprender uma nova linguagem denominada nesC, sendo esta uma extensão da linguagem de programação C. Além de ser uma nova linguagem, o nesC foi criado incorporando as características do TinyOS como os conceitos e modelo de execução, e ainda obedece um novo paradigma (orientado a eventos).

Em [GAY et al., 2003; GAY et al, 2007] são descritos os desafios que se tornaram prioridades no desenvolvimento da linguagem nesC, sendo os principais desafios na construção de aplicações para RSSF os seguintes:

- Robustez: Assim que a rede for instalada no ambiente, ela deve permanecer funcionando sem intervenção humana, por meses ou até anos.
- Baixo consumo de recursos: Os nós sensores possuem um hardware limitado, e dentro dessa limitação estão inclusas uma baixa quantia de memória RAM e uma bateria com vida útil limitada.
- Diversas implementações de serviços: As aplicações devem poder escolher entre múltiplas implementações de serviços, como, por exemplo, roteamento *multihop*.
- Evolução do hardware: O *hardware* dos nós sensores estão sempre evoluindo; As aplicações e a maioria dos serviços devem ser portáveis pelas gerações de *hardware*.
- Adaptabilidade aos requerimentos das aplicações: As aplicações têm diversos requerimentos diferentes em termos de tempo de vida, comunicação, sensoriamento, etc.

Os programas em nesC são formados por componentes, que podem ser combinados para formar os aplicativos, de modo que a modularidade de reusabilidade do código é alta. Na linguagem nesC, o comportamento de um componente é especificado por um conjunto de interfaces bem definidas. Estas interfaces são bidirecionais, e devem informar o que o componente usa e o que ele provê. Na Figura 7.11 é mostrado um componente (Timer) e sua interface.

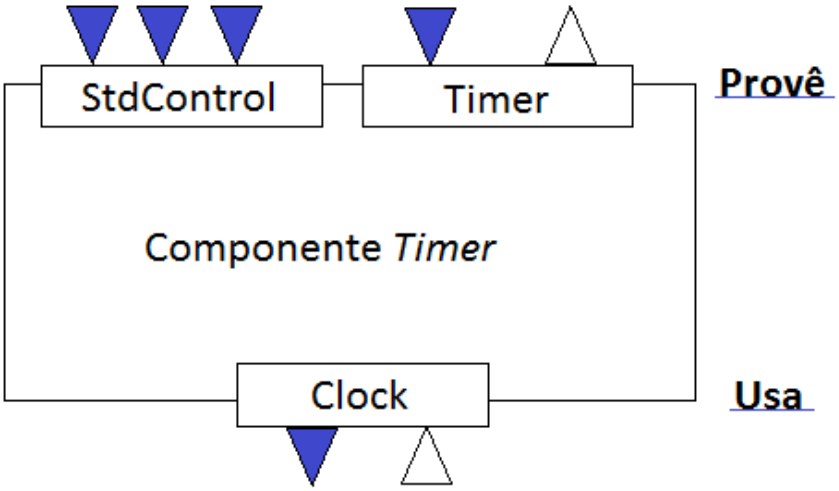


Figura 7.11 Um Componente e sua Interface

Componentes possuem algumas características parecidas com os objetos, de tal modo que também encapsulam estados e interagem por interfaces bem definidas, mas diferentemente dos objetos, não ocorre herança entre os componentes, além de não existir alocação dinâmica.

A dificuldade encontrada no desenvolvimento dos componentes não está vinculada à escrita do código, pois a linguagem nesC é muito parecida com a linguagem C, mas em vincular os vários componentes do código no intuito de formar o programa completo.

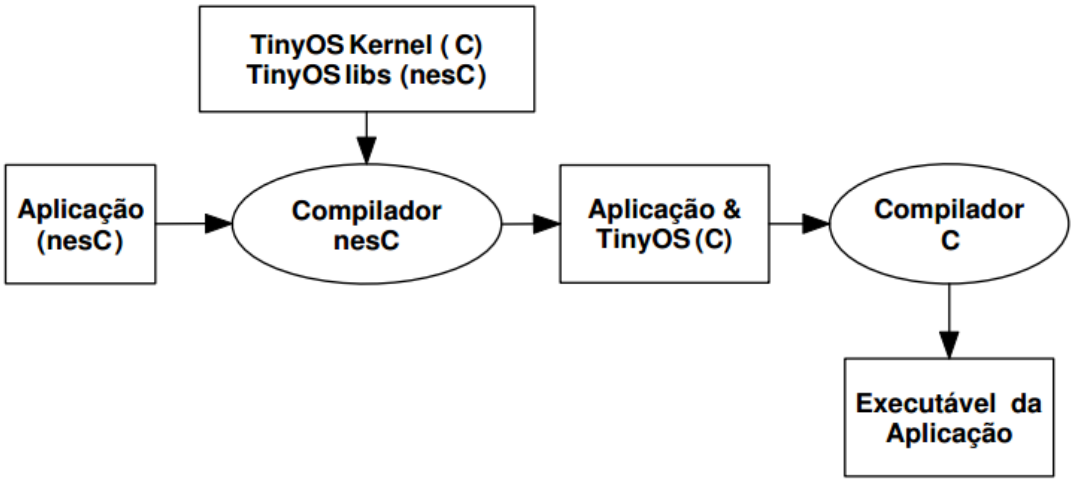


Figura 7.12 Etapas Envolvidas no Desenvolvimento de uma Aplicação no TinyOS

O processo de desenvolvimento de uma aplicação no TinyOS é feito em etapas, sendo as mesmas mostradas na Figura 7.12. A aplicação a ser desenvolvida é descrita na linguagem de programação nesC. Na etapa seguinte o código criado para a aplicação somado ao kernel do TinyOS e às bibliotecas do mesmo são compilados no compilador nesC, formando o código da aplicação com TinyOS em C. O código gerado é então compilado por um compilador C, resultando em um executável da aplicação. Em cada etapa do desenvolvimento os arquivos gerados possuem extensões diferentes que facilitam a identificação do tipo do arquivo. Na Figura 7.13 pode-se acompanhar a geração dos arquivos e suas extensões. A extensão para os arquivos em nesC é `.nc`. Quando o arquivo do nesC é compilado pelo compilador nesC (`ncc`) são gerados arquivos em C com a extensão `.c`. Então um compilador C (neste caso o `avr-gcc`) irá gerar um código de máquina com a extenção `.s` que será utilizando como entrada para um montador (neste caso o `avr-as`). Como saída o montador gera o código objeto que possuí extensão `.o`. E concluindo o processo de compilação o compilador `avr-gcc` cria o código executável com a extensão `.exe`. Para ser carregado no hardware do Mica Motes o arquivo executável `.exe` ainda deve ser convertido para o formato S-Records com extensão `.srec`, conversão esta feita pelo `avr-objcopy`.

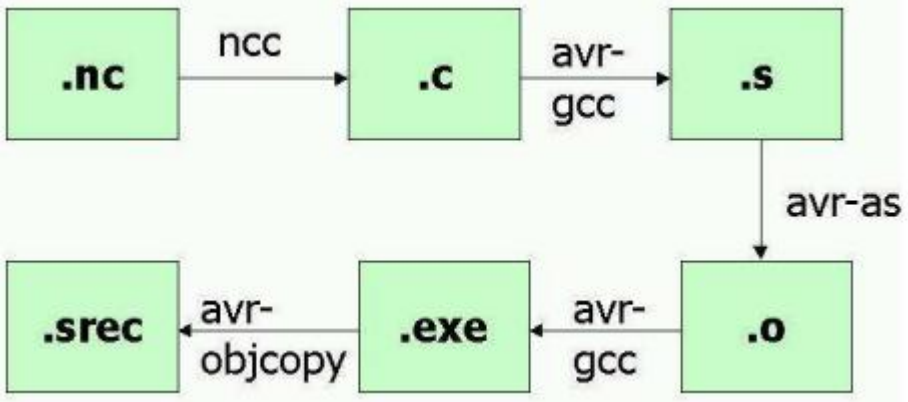


Figura 7.13 Etapas de Compilação e Extensão dos Arquivos Gerados

Em [GAY et al., 2003] são descritos os conceitos básicos da linguagem nesC, sendo eles:

- Separação de construção e composição: Os programas são constituídos pela união de elementos denominados **componentes**.
- Especificação do comportamento de componentes através de um conjunto de interfaces: Interfaces podem ser fornecidas pelos componentes, onde está sendo representada a funcionalidade que se deseja fornecer ao usuário, ou ainda podem ser utilizadas pelos componentes, quando se está seguindo uma funcionalidade que precisa pra realizar sua tarefa.
- Interfaces são bidirecionais: Funções podem ser implementadas pelo fornecedor da interface (Comandos) ou podem ser implementadas pelo usuário da interface (eventos).
- Componentes são ligados por meio das interfaces: Recurso que melhora a eficiência em tempo de execução, e ainda permite uma melhor análise estática dos programas.
- O modelo de concorrência no nesC é baseado em tarefas "*run-to-completion*"; e gerenciadores de interrupção, os quais tem o poder de interromper tarefas ou mesmo outras interrupções.

7.21.1 Componentes e Interfaces

O nesC possui dois tipos de componentes, sendo eles:

- **Módulos.** Componentes que são implementados com código na linguagem nesC.
- **Configurações.** Componentes que são implementados pela ligação entre os componentes.
- **Interfaces**. São componentes que devem ser implementados nos módulos.

7.22. Instalação do ambiente de Desenvolvimento

Para trabalhar com o TinyOS são necessários basicamente:

- O código-fonte do TinyOS
- O compilador nesC
- O compilador gcc específico para a plataforma de Hardware
- Bibliotecas

O processo de compilação de uma aplicação é feito utilizando o comando *make*, que é um utilitário muito popular nos ambientes *unix-like* para automatizar tarefas.

Neste caso é utilizado o sistema operacional Ubuntu Linux versão 8.04 como base para a instalação do ambiente de desenvolvimento do TinyOS assim como para o desenvolvimento das aplicações.

A instalação do ambiente é simples e pode ser realizada em três passos:

Antes de tudo deve-se remover qualquer repositório antigo do TinyOS, que por ventura esteja no arquivo /etc/apt/sources.list.

- Adicionar o repositório do tinyOS na lista de pacotes do sistema, editando o arquivo sources.list:

```
cd /etc/apt
```

```
edit sources.list
```

- Inserir a linha a seguir no final do arquivo sources.list

```
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu
lucid main
```

- Atualize o conteúdo do gerenciador de pacotes com o comando:

```
sudo apt-get update
```

- Instale a ultima versão do tinyOS. Neste caso vamos utilizar a ultima versão disponível, a tinyos-2.1.1.

```
sudo apt-get install tinyos-2.1.1
```

- Aceite o conteúdo das duas permissões, e edite o CLASSPATH do Java com o seguinte comando:

```
edit /opt/tinyos-2.1.1/tinyos.sh
```

- Encontre a linha:

```
export CLASSPATH=$TOSROOT/support/sdk/java
```

- Troque ela pela seguinte:

```
export
CLASSPATH=$TOSROOT/support/sdk/java/tinyos.jar:.
```

- Para concluir, edite o arquivo .bashrc, fazendo:

```
edit ~/.bashrc
```

- No final do arquivo, insira a linha a seguir:

```
source /opt/tinyos-2.1.1/tinyos.sh
```

- A instalação já está realizada, e deve-se dar permissão de escrita para a pasta /opt/tinyos-2.1.1/, com o seguinte comando:

```
sudo chmod 775 -R ./tinyos-2.1.1/*
```

Obs.: Onde está escrito **edit** deve ser substituído pelo editor de texto a ser utilizado e de sua preferência, como o VIM ou o GEDIT.

7.23. O Simulador TOSSIM

Segundo [RUIZ et al., 2004] o TOSSIM (TinyOS SIMulator) é um simulador para eventos para RSSF que utilizam o TinyOS. Em vez de compilar o programa para ser utilizado em um nó sensor, o programador compila o mesmo para o ambiente do TOSSIM, desta forma podendo executar o código no próprio computador executando o TOSSIM. Desta maneira podendo deputar, testar, e analisar o código gerado em ambientes controlados. Devido ao TOSSIM ser executado no computador, podem ser utilizadas ferramentas de desenvolvimento para examinar os códigos.

O TOSSIM tem como objetivo ser um meio fiel para simulação das aplicações voltadas para o TinyOS. Como o TOSSIM pode ser utilizado para tentar entender as causas do mundo real, os resultados gerados por ele, não devem ser levados como uma avaliação absoluta, pois ele não consegue gerar todas as situações que podem ocorrer no mundo real. O TOSSIM tem a capacidade de simular ambientes com poucas ou até milhares de unidades dos nós de sensoriamento em simultâneo. Durante a simulação, cada nó faz uso do mesmo programa TinyOS.

7.24. Desenvolvendo uma aplicação em nesC

Nesta seção é explicado um código genérico em nesC. Como já foi visto, todo programa em nesC é formado por componentes. A seguir estão descritos os principais componentes básicos.
Qualquer aplicação em nesC necessita ao menos dos seguintes arquivos:

- Makefile
- Configuração
- Módulo

Cada um destes representa um arquivo, e todos eles estão armazenados no mesmo diretório. Como representado na Tabela 7.3.

Tabela 7.3 Estrutura Básica de Arquivos

Nome do Arquivo	Descrição
Makefile	Makefile
testC.nc	Arquivo de Configuração
testP.nc	Arquivo do Módulo
test.nc	Arquivo de Interface
test.h	Arquivo de Cabeçalho

É possível fazer uso de qualquer nome para os arquivos, entretanto é recomendado utilizar esta convenção para nomear os arquivos, de modo que tudo que seja feito possa ser entendido por outros que venham a utilizar os arquivos posteriormente. No endereço http://www.tinyos.net/tinyos-2.x/doc/html/tep3.html pode ser encontrada esta, além de outras boas práticas para o TinyOS. Para este caso estaremos utilizando o sufixo (não é a

extensão) C no final dos nomes dos arquivos de configuração e o sufixo M no final dos nomes dos arquivos de módulos. Nas aplicações de exemplo do TinyOS estas nomenclaturas não são utilizadas.

7.24.1. Arquivo Makefile

A estrutura básica do arquivo Makefile é:

```
COMPONENT=testC
include $(MAKERULES)
```

Em **COMPONENT** é indicado o nome do arquivo de configuração utilizado (sem a extensão), sendo neste caso o arquivo *testC*. O compilador terá o papel de buscar no mesmo diretório o arquivo de configuração indicado.

7.24.2. Arquivo de Configuração

Como está indicado no arquivo *Makefile*, o arquivo de Configuração está nomeado como *testC.nc*:

```
configuration testC
{
}
implementation
{
  components MainC;
  components testP as App;
  App -> MainC.Boot;
}
```

A Configuração contem duas seções, a seção **configuration** e a seção **implementation**:
configuration: Não faz nada neste caso.
implementation: Realiza a ligação com outros componentes, neste caso estão sendo declarados os componentes que estão sendo utilizados nessa configuração. Também é obrigatório a todos os arquivos de configuração que estes inicializem os seus módulos, neste caso testM. No exemplo é informado ao compilador que a interface *Boot* do MainC será utilizada pelo módulo testM.
Obs.: A palavra-chave **as** é utilizada para renomear componentes dentro do código.

7.24.3. Arquivo de Modulo

O arquivo de Módulo é o *testP.nc*:

```
module testP ()
{
  uses interface Boot;
}
implementation
{
  event void Boot.booted()
```

```
    {
    }
}
```

Da mesma maneira que a Configuração, o Módulo também é composto de duas seções, sendo elas **module** e **implementation**:
module: Com a palavra **uses** se indica a interface a ser utilizada. Caso se deseje checar, o sistema provê a interface *Boot* (*TOSROOT/tos/interfaces/Boot.nc*), pode-se notar que esta interface em particular oferece o evento **booted**.
implementation: Nas aplicações principais, neste local são feitas as chamadas de comandos e é onde os eventos acontecem. Neste exemplo apenas um evento acontece (*booted*). A sintaxe para os eventos é sempre a mesma, primeiro se indica o nome da interface (neste caso *Boot*) e em seguida um ponto e depois o nome do evento (neste caso *booted*) como segue o modelo: "*NomeInterface.Evento*".

Esta é uma aplicação compilável, mas que não faz nada. Ela tem apenas fins explicativos. Para ver uma aplicação com funcionalidade, existe o **Blink**. A seguir será analisado do código do **Blink**.

7.24.4. O Blink

No aprendizado da maioria das linguagens de programação, sempre se inicia pela aplicação mais básica da linguagem, que costuma ser o *Hello World*. No caso da linguagem nesC não existe um *Hello World*, mas em contrapartida existe um programa de teste que se caracteriza por ser o primeiro programa no aprendizado do nesC, este programa é o Blink. O conteúdo desta seção é baseado no tutorial disponibilizado no site do TinyOS (*http://docs.tinyos.net/tinywiki/index.php/Getting_Started_with_TinyOS*) e baseado no modelo de sensor *mica2* da empresa Crossbow (*http://www.xbow*.com) como plataforma de desenvolvimento.

Figura 7.14 Modelo de sensor mica2 da Crossbow

O Blink é uma aplicação bastante simples usada para entender a estrutura de um programa em nesC. A palavra Blink é de origem inglesa e significa *Piscar*, desta maneira a função do Blink é fazer os três LED's do nó sensor piscarem cada um com uma frequência e ordem definidas:

Tabela 7.4 Frequência dos piscar dos LED's na aplicação Blink

LED	Frequência em Hz
0	25Hz
1	5Hz
2	1Hz

Como visto nas seções anteriores, as aplicações em nesC são compostas por componentes, e no caso do **Blink** são dois: um módulo denominado *BlinkC.nc* e uma configuração chamada *BlinkAppC.nc*. O *BlinkAppC.nc* é utilizado para fazer a ligação entre o módulo *BlinkC.nc* e os outros módulos necessários à aplicação. Vale citar que ao dividir o código em configurações e módulos, a linguagem tem uma vantagem que é o reuso de componentes, apenas realizando a ligação entre objetos existentes.

7.24.5. Arquivo de Configuração do Blink (BlinkAppC.nc)

A seguir, está o código do arquivo *BlinkAppC.nc*, para analise:

```
configuration BlinkAppC {
}
implementation {
  components MainC, BlinkC, LedsC;
```

```
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as Timer1;
  components new TimerMilliC() as Timer2;

  BlinkC -> MainC.Boot;
  BlinkC.Timer0 -> Timer0;
  BlinkC.Timer1 -> Timer1;
  BlinkC.Timer2 -> Timer2;
  BlinkC.Leds -> LedsC;
}
```

No início do arquivo aparece a palavra **configuration**, ela é utilizada para descrever que o arquivo é do tipo configuração:

```
configuration BlinkAppC {
}
```

Neste trecho do código não há conteúdo entre as chaves, sendo possível e se necessário especificar *uses* e *provides*, respectivamente para usar ou prover alguma interface dentro da aplicação.

A configuração é realmente feita dentro do espaço destinado à implementação, definido no campo **implementation**. Neste local ficam as declarações dos componentes a serem utilizados no código, para o Blink são declarados os componentes *MainC*, *BlinkC*, *LedsC* e *TimerMilliC*, este ultimo sendo instanciado três vezes, e sendo referenciado por *Timer0*, *Timer1* e *Timer2*.

Nas quatro ultimas linhas deste arquivo é feita a ligação entre as interfaces que o arquivo BlinkC usa e as interfaces providas pelos componentes *LedsC* e *TimerMilliC*.

7.24.6. Arquivo do Módulo do Blink (BlinkC.nc)

```
  module BlinkC {
  uses interface Timer<TMilli> as Timer0;
  uses interface Timer<TMilli> as Timer1;
  uses interface Timer<TMilli> as Timer2;
  uses interface Leds;
  uses interface Boot;
}
implementation
{
  // implementation code omitted
}
```

No início deste arquivo há a indicação de este é um módulo denominado BlinkC, isto é feito na primeira linha com a descrição **module**. Em seguida são descritas as interfaces que o módulo faz uso, sendo elas três interfaces *Timer<TMilli>*, e as interfaces

Leds e *Boot*. Neste trecho deve existir coerência entre as interfaces que estão sendo definidas neste trecho do código, com as ligações realizadas no arquivo de configurações *BlinkAppC.nc*.

Para exemplificar a coerência que deve existir entre o arquivo de configuração e o de módulo, basta notar a interface *Leds*, o arquivo de configuração usa a interface, ou seja, ele irá utilizar algum comando disponibilizado por esta, mas as interfaces não definem como o comando será implementado, esta é uma tarefa para o programador. No arquivo de configurações deve ser definido qual componente implementa a interface. Neste exemplo, a interface *Leds* no arquivo de configuração *BlinnkAppC.nc* está conectada à interface Leds do componente LedsC:

```
BlinkC.Leds -> LedsC;
```

A linguagem nesC faz uso da flecha (->) para realizar a ligação entre interfaces e componentes. Quando se tem algo semelhante a (Inter -> Comp) deve-se ler que Inter está conectado a Comp ou ainda Inter usa Comp. Nós casos em que um componente usa ou provê várias interfaces é interessante nomear as interfaces para facilitar o trabalho. Neste exemplo do Blink o arquivo BlinkC usar três instâncias de Timer, sendo elas *Timer0*, *Timer1* e *Timer2*. Nos casos em que um componente tiver apenas uma instância de uma interface, é possível suprimir o nome da interface da seguinte maneira:
Em vez de usar:

```
BlinkC.Leds -> LedsC.Leds
```

Utilizar somente:

```
BlinkC.Leds -> LedsC;
```

7.24.7. Compilando a aplicação

Para realizar o processo de compilação da aplicação de uma aplicação TinyOS é necessário ter instalado o TinyOS como descrito nas seções anteriores, tendo ele instalado, basta ir no diretório da aplicação e fazer uso do comando **make**. A sintaxe a ser utilizada é ***make [plataform]***, sendo *plataform* a representação do hardware a ser utilizado no sensor. Neste caso está sendo utilizado o modelo de sensor mica2. Logo deve-se compilar da seguinte maneira: ***make mica2***.

O seguinte conteúdo deverá ser impresso na tela:

```
mkdir -p build/mica2
    compiling BlinkAppC to a mica2 binary
ncc -o build/mica2/main.exe -Os -finline-limit=100000 -Wall -
Wshadow        -Wnesc-all        -target=mica2        -fnesc-
cfile=build/mica2/app.c            -board=micasb            -
DIDENT_PROGRAM_NAME=\"BlinkAppC\"  -DIDENT_USER_ID=\"marvin\"
-DIDENT_HOSTNAME=\"skyship\"    -DIDENT_USER_HASH=0x3b3023baL
-DIDENT_UNIX_TIME=0x481a6a0eL    -DIDENT_UID_HASH=0xe1b7fae7L
-fnesc-dump=wiring      -fnesc-dump='interfaces(!abstract())'
-fnesc-dump='referenced(interfacedefs,     components)'     -
fnesc-dumpfile=build/mica2/wiring-check.xml  BlinkAppC.nc  -lm
compiled BlinkAppC to build/mica2/main.exe
```

```
            7130 bytes in ROM
              52 bytes in RAM
avr-objcopy     --output-target=srec     build/mica2/main.exe
build/mica2/main.srec
avr-objcopy     --output-target=ihex     build/mica2/main.exe
build/mica2/main.ihex
    writing TOS image
```

A compilação foi efetuada com sucesso.

7.25. Conclusão

Redes de Sensores Sem Fio (RSSF) é um campo que ainda tem muito a ser explorado, por ser relativamente novo e por ter várias questões a serem otimizadas, como a principal delas que é o consumo energético da rede. Existem muitos desafios em encontrar soluções otimizadas para as problemáticas desse tipo de rede, visto que não se pode realizar o reaproveitamento dos algoritmos utilizados nas redes tradicionais, desta forma tornando necessário o empenho para se criar algoritmos novos e voltados exclusivamente para atender às necessidades e diminuir as barreiras dessa nova tecnologia que tem muito a oferecer.

Como visto o TinyOS é um sistema operacional feito exclusivamente para as Redes de Sensores e projetado baseado nas dificuldades das mesmas, este sistema tem como característica básica a programação orientada a componentes e eventos.

O nesC, linguagem utilizada para programar para TinyOS é baseada na linguagem C e é simples, além de trabalhar com módulos, o que permite o reuso de componentes, reduzindo a quantidade de código a ser digitado.

Enfim, este trabalho tem como função apresentar o mundo das Redes de Sensores Sem Fio falando de como funciona, e detalhando superficialmente a mesma, além de mostrar áreas em que a tecnologia pode ser aplicada e ainda tratar do TinyOS, sistema voltado para a área de RSSF, concluindo com um demonstrativo de uma aplicação básica.

7.26. Referências

AKYILDIZ, I. F., et al. A Survey on Sensor Networks. IEEE Communications Magazine, 2002, 40(8). p 102–114.

BERNDT, A. Introdução a Redes de Sensores sem Fio (RSSF), Departamento de Ciência da Computação - Universidade do Estado de Mato Grosso (UNEMAT) 2008.

BHATTACHARYA, S., et al. Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks. MobiSys 2003: The First International Conference on Mobile Systems, Applications, and Services, p. 173-185, USENIX, 2003.

BRITO, L. F. M. N. Análise Comparativa da Vazão e do Consumo de Energia em Redes de Sensores Sem Fio com Topologia Estrela Utilizando o Padrão IEEE 802.15.4 no Modo *BEACON-ENABLED* e *NONBEACON*. (Trabalho de Conclusão de Curso em Engenharia de Redes de Comunicação) – Universidade de Brasília, Dez. 2009.

DELICATO, F. Middleware Baseado em Serviços Para Redes de Sensores Sem Fio. (Tese de Doutorado em Ciências em Engenharia Elétrica) – Universidade Federal do Rio de Janeiro, Jun. 2005.

DULMAN, S., et al. Introduction to wireless sensor network, Embeded systems Handbook, p. 31.1 – 31.10, 2006.

FAÇANHA, T. S. Rede de Sensores Sem Fio: Uma Abordagem para Detecção de Falhas em Sistemas Elétricos. Dissertação (Mestrado) - Centro Federal de Educação Tecnológica do Ceará: Curso Superior de Tecnologia em Telemática, Fortaleza - CE, 2007.

GAY, D., et al. Software Design Patterns for TinyOS. Em ACM Trans. Embedd. Comput Syst. 6, 4, Article 22 (2007), 39 páginas.

GAY, D., et al. The NesC Language: A Holistic Approach to Networked Embedded System. San Diego, California, USA, 2003.

GAY, D. et al.. NesC 1.1 Language Reference Manual. 2003.

JABOUR, E. C. M. G. Um Protocolo de Transporte Colaborativo para Redes de Sensores Sem Fio. (Tese de Doutorado em Engenharia Elétrica) – Universidade Federal do Rio de Janeiro, Mar. 2009.

KOUSHANFAR, F., Potkonjak, M., Sangiovanni-Vincentelli, A.. Fault-Tolerance in wireless sensor networks. In Handbook of Sensor Nerworks. CRC press, 2004.

LOUREIRO, A. A. F., et al. Redes de Sensores Sem Fio. UFMG Belo Horizonte. 2003.

MACEDO, D. F., et al. Avaliando Aspectos de Tolerância a Falhas em Protocolos de Roteamento para Redes de Sensores Sem Fio, 2005.

NAKAMURA, F. G. Planejamento dinâmico para controle de cobertura e conectividade em redes de sensores sem fio planas. Dissertação (Mestrado) - Departamento de pós-graduação em Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte - MG, 2003.

NETO, F. C. Semsus: Middleware para Redes de Sensores Sem Fio Baseado em Serviços Semânticos. Dissertação (Mestrado) – Universidade do Estado do Rio Grande do Norte: Curso de Mestrado em Ciência da Computação, Mossoró - RN, Jan. 2010.

RUIZ, L. B., et al. Arquiteturas para Redes de Sensores Sem Fio (SBRC 2004), 2004.

SANTOS, J. A. G. Estudo do Balanceamento de Carga e Proposta de Uso em Redes de Sensores Sem Fio Hierárquicas. (Trabalho de Conclusão de Curso em Ciência da Computação) – Universidade do Estado do Rio Grande do Norte, Fev. 2008.

SILVA, R. C. Redes de Sensores Sem Fio. (Trabalho de Conclusão de Curso em Ciências da Computação) – Universidade Estadual de Montes Claros, Jun. 2006.

SOARES, L. C. Redes de Sensores Sem Fio com Enfoque em Protocolos de Comunicação. UEL Londrina. 2007.

VELLOSO, P. B., et al. Uma Analise da Capacidade de Transmissão de Voz em Redes ad hoc. UERJ, 2003.

VIEIRA, M. A. M. et al., Como Obter o Mapa de Energia em uma Rede de Sensores Sem Fio? Uma Abordagem Tolerante a Falhas. UFMG, 2003.

VIEIRA, L. F. M. Middleware para Sistemas Embutidos e Redes de Sensores. Dissertação (Mestrado) – Universidade Federal de Minas Gerais: Curso de Mestrado em Ciência da Computação, Belo Horizonte - MG, Abr. 2004.

Capítulo 8

Desenvolvimento de Aplicativos para Android

Dyego Carlos Sales de Morais, Ivan Fillipe Rodrigues dos Santos

Abstract

In this mini-course will introduce the Android platform, a platform on the rise in development applications for mobile devices. In this mini-course will show the steps required for installing and configuring the development environment, presentation of the file structure, demonstrating the use of the main components of layout, implementation of concepts, and finally execution of an example mobile device.

Resumo

Nesse minicurso iremos apresentar a plataforma Android, uma plataforma em ascensão no desenvolvimento de aplicativos para dispositivos móveis. Nesse minicurso serão mostrados os passos necessários para a instalação e configuração do ambiente de desenvolvimento, apresentação da estrutura de arquivos, demonstração do uso dos principais componentes de layout, implementação de conceitos e por fim execução de um exemplo no dispositivo móvel.

8.1 Dados Gerais

8.1.1. Objetivos do Curso

O minicurso tem por objetivos capacitar os alunos ao desenvolvimento de aplicações para dispositivos móveis, através da apresentação e implementação de conceitos inerentes a plataforma Android, bem como execução e instalação de aplicativos em um dispositivo móvel real.

8.1.2. Tipo do Curso

O minicurso tem sua carga-horária dividida entre apresentação de conceitos teóricos da plataforma Android e demonstração de práticas de codificação. Importante salientar que como serão abordados muitos conceitos teóricos, a prática do curso terá menor carga-horária, porém o suficiente para capacitar os alunos a desenvolver, testar e instalar uma aplicação básica para seus dispositivos.

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

8.1.3. Tratamento dado ao tema

Inicialmente será realizada uma visão geral sobre o estado da arte no desenvolvimento de aplicações para dispositivos móveis. Em seguida será apresentado um comparativo entre as principais tecnologias de desenvolvimento para plataformas móveis, em especial Android. Por fim, será demonstrada a codificação de alguns conceitos na prática, incluindo instalação em dispositivo real.

8.1.4. Perfil desejado dos participantes

O minicurso é melhor aproveitado por desenvolvedores (profissionais ou estudantes) que dominam o paradigma orientado a objetos, de preferência Java.

8.1.5. Infra-estrutura física necessária para a apresentação

Projetor multimídia; quadro-branco; pincéis; apagador.

8.2 Estrutura prevista detalhada do texto

Tecnologias móveis produzem vários benefícios para as diversas áreas do conhecimento. Em todas as áreas tem-se uma van*tag*em na utilização desse tipo de tecnologia: simplificação do acesso e da atualização de informações. Em saúde, por exemplo, é possível obter uma diminuição de erros e inconsistências se dispositivos móveis forem utilizados estrategicamente para esse objetivo. E em educação, com a utilização de aparelhos portáteis pode-se auxiliar no processo de construção do conhecimento. Na 15ª Bienal do Livro, o Ministro da Educação brasileiro, Fernando Haddad, anunciou que o governo vai distribuir *tablets* para alunos de escolas públicas a partir de 2012 com o objetivo de universalizar o acesso à tecnologia.

Dispositivos móveis são pequenos computadores com tela menor e mecanismo de entrada de dados limitado quando comparados a computadores *desktop*. Esses aparelhos costumam ter a capacidade de conexão via rede sem fio e possuem como principal premissa, a permissão de execução de algumas tarefas enquanto o operador se movimenta. Já uma aplicação computacional móvel é um pedaço de software embutido em um dispositivo móvel [B'FAR 2004] [LEE et al. 2004].

O poder computacional de dispositivos móveis está em intenso avanço e já se pode compará-lo ao dos microcomputadores de mesa, visto que já possuem inclusive processadores *multi-core*. Ademais, novos elementos de hardware nos dispositivos móveis já são comuns, como:

- acelerômetro: funcionalidade que calcula a aceleração utilizada em discos rígidos parando o HD em caso de queda, além de ser, atualmente, implantado em diversos dispositivos móveis, produzindo uma nova interação com a máquina;
- GPS: sistema de posicionamento global, popularmente conhecido por GPS (acrónimo do original inglês *Global Positioning System*, ou do português "geo-posicionamento por satélite") é um sistema de navegação por satélite que fornece a um aparelho receptor móvel sua posição;
- *display touch-screen*: telas sensíveis ao toque, ou ecrã que é capaz de reconhecer o toque na tela como entrada de dados;

- entre diversos outros.

Desse modo já é possível usufruir de aplicações em *smartphones* e *tablets* com funcionalidades que seriam inviáveis se aplicadas em computadores *desktop*. Essa convergência de tecnologias em um único dispositivo faz dos aparelhos móveis, em especial os *smartphones*, a nova tendência do mercado tecnológico.

Segundo a Anatel (2012) o Brasil possui mais de 188 milhões de aparelhos celulares, o que resulta numa média de quase um celular por habitante. Em 2010, as vendas de dispositivos móveis cresceram 31,8% em todo o mundo, somando aproximadamente 1,6 bilhões de unidades vendidas entre celulares e *smartphones* [Gartner 2011].

A ascensão de tecnologias como as plataformas Android e iOS, vem mudando o cenário de dispositivos e tecnologias utilizadas para aplicações móveis. Isso vem ocorrendo, pois juntamente com o obsoleto e descontinuado, porém ainda dominante, sistema operacional SymbianOS, essas novas tecnologias já estão presentes na maioria dos dispositivos móveis [StatCounter 2012].

Outro dado relevante é que em se tratando apenas de *smartphones*, o Android já é o sistema móvel mais utilizado no mundo [InfoGraphic Labs 2012]. Dentre os aparelhos mais vendidos, os sistemas operacionais mais utilizados são: Symbian; iOS; Android; e BlackBarry (figura 8.1).

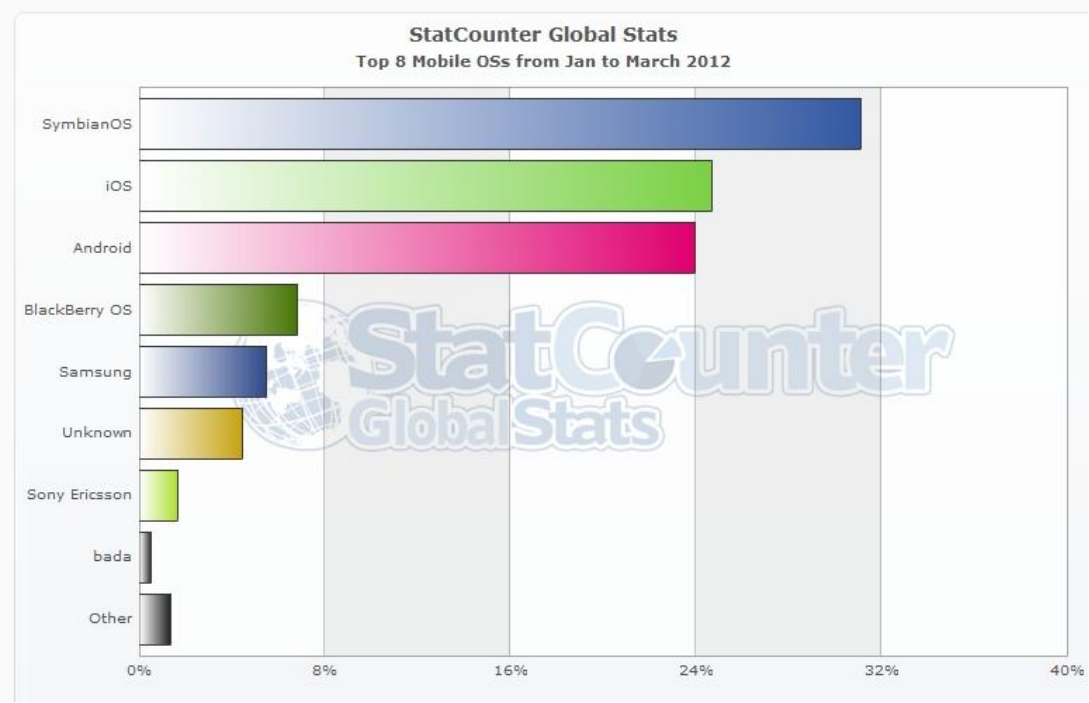


Figura 8.1. Sistemas operacionais mais utilizados em dispositivos móveis no primeiro trimestre de 2012 [StatCounter 2012].

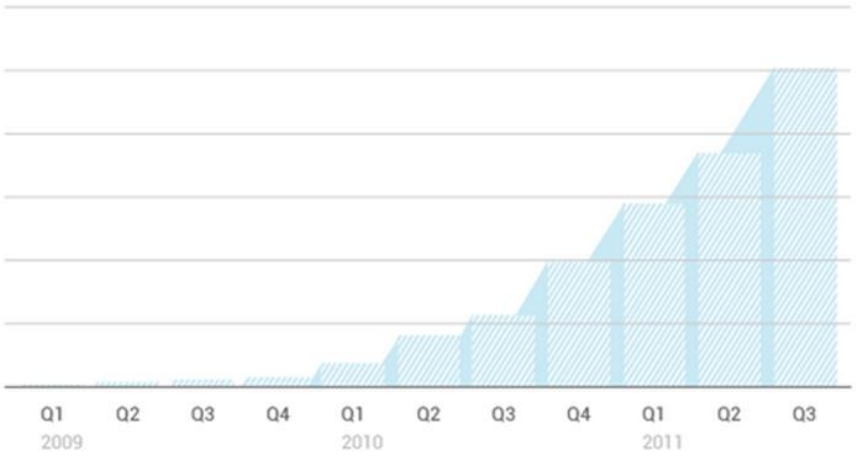
Dentre os 4 líderes de mercado de sistemas operacionais móveis apresentados na figura 8.1, apenas um (iOS) não tem suporte a Java. A van*tag*em da utilização do Java se dá por uma série de fatores, dentre os quais se destacam: portabilidade, por suportar uma variedade de dispositivos com diferentes níveis de recursos; flexibilidade, por oferecer funcionalidades específicas dos diversos dispositivos; uniformidade, para manter uma

arquitetura comum; além de ser orientada a objetos, que permite código pouco acoplado e altamente coeso.

As principais características do Android são:

- *Application framework*: proporciona a reutilização e substituição de componentes;
- Dalvik *virtual machine*: otimizada para dispositivos móveis;
- *Browser* Integrado: baseado no *webkit engine*;
- Gráficos Otimizados: (aceleração de hardware é opcional);
- SQLite: um poderoso e leve *engine* de banco de dados relacional disponível para todas as aplicações
- Suporte multimídia para áudio, vídeo e formatos de imagem (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF);
- Telefonia GSM (dependente de hardware);
- Bluetooth, EDGE, 3G, e Wi-Fi (dependente de hardware);
- Câmera, GPS, compasso, e acelerômetro (dependente de hardware);
- Rico ambiente de desenvolvimento, incluindo um emulador de dispositivo, ferramentas de depuração, memória, performance e um *plugin* para o Eclipse (ADT).

Ademais, outros dados relevantes sobre o Android são que ele possui mais de 200 milhões de dispositivos ativados, com uma média de 550 mil ativações diárias; está presente em 137 países e regiões; e, conforme mostra gráfico apresentado na figura 8.2, apenas no terceiro trimestre de 2011, os usuários baixaram 2,4 bilhões de aplicações para os seus dispositivos móveis com Android [Google 2011].



Growth in app consumption. Users downloaded more than 2.4 billion apps from Android Market in Q3 2011 alone.

Figura 8.2. Gráfico de download de aplicativos para Android até o terceiro trimestre de 2011 [Google 2011].

Estes dados mostram o potencial do mercado de aplicações móveis, bem como os benefícios que se pode ter ao dominar conceitos de programação para dispositivos móveis com Android.

Diante deste cenário, o curso de Desenvolvimento de Aplicativos para Android visa capacitar, qualificar e potencializar mão-de-obra para o mercado regional no desenvolvimento de aplicações para essa plataforma. O conteúdo programático oferecido pelo curso está estruturado da seguinte forma:

8.2.1 Introdução ao Desenvolvimento de Aplicações Móveis

Ultimamente diversas plataformas para desenvolvimento vem surgindo com foco em sistemas para celulares e *smartphones*. Dentre estas plataformas se destacam: SymbianOS; Cocoa (*framework* do iPhone); BlackBarry; Android.

O Symbian foi o primeiro sistema operacional (SO) para celulares, criado pela Psion para a série Palm [Nokia 2010]. A Nokia iniciou a utilização da plataforma Symbiam 60 em seus smartphones em 2002. O Symbiam possui suporte a serviço de mensagens multimídia (do inglês, *Multimedia Messaging Service* – MMS), Bluetooth, *wireless*, infravermelho, dentre outros. Quanto a desempenho, o Symbian realiza um bom consumo de memória, energia, processamento, entre outros fatores essenciais. Permite também, o uso de várias linguagens de programação, tais como: Symbian C/C++, Java ME, Python, entre outras. Além disso, há diversas ferramentas de desenvolvimento para Symbiam que são distribuídas gratuitamente.

O iPhone é o *smartphone* da Apple, que utiliza como plataforma de desenvolvimento uma SDK exclusiva o Cocoa, que utiliza a linguagem de programação Objective-C, muito utilizada na plataforma MAC [Apple 2010]. O custo para utilização desta SDK é de U$99 dólares anuais. Além disso, as aplicações desenvolvidas para o iPhone são disponibilizadas somente na loja da Apple, a AppStore. Entretanto, qualquer softwatre desenvolvido para o iPhone antes de ir para AppStore passam por uma revisão da Apple. A SDK do iPhone roda somente em sistemas computacionais Mac.

O BlackBerry é o SO para *smartphones* BlackBerry da empresa Research in Motion – RIM. O BlackBerry utiliza o Java como linguagem de desenvolvimento e é compatível com aplicações J2ME. Entre seus maiores feitos destaca-se sua ferramenta de sincronização de *e-mails*. Assim que o servidor recebe o *e-mail* este é enviado para o dispositivo [BlackBarry 2010].

Inicialmente, o Android foi desenvolvido pela Google, mas atualmente é gerido pela *Open Handset Alliance* – OHA, um grupo formado por mais de 30 empresas, dentre as quais destacamos Google, Motorola, HTC, LG, Sony Ericsson, Toshiba, Sprint Nextel, China Mobile, T-Mobile, ASUS, Intel, dentre outros. O Android tem sua distribuição feita pelo Android Market, com o custo de US$25 dólares de registro. Apesar de ter sido desenvolvido inicialmente para *smartphones*, hoje ele roda em diversas outras plataformas

como *tablets* e *netbooks*, e seu uso tem se expandido cada vez mais em dispositivos portáteis e sistemas embarcados em geral, inclusive relógios. No subtópico seguinte é abordado mais sobre esse SO e sua arquitetura.

8.2.2. Introdução ao Desenvolvimento de Aplicações Móveis

A plataforma de desenvolvimento Android possui as seguintes principais características: possui grande parte de seu código livre; as aplicações do Google já vêm integradas no sistema do Android; e permite que com uma mesma conta seja possível acessar uma série de ferramentas: *Email*, *chat*, *sites*, *docs*, grupos, calendários, etc; além de utilizar uma máquina virtual (Dalvik) projetada para aperfeiçoar a memória e os recursos de hardware em um dispositivo móvel (Figura 8.3)[Android 2010].

No desenvolvimento do SO foi utilizada a versão 2,6 do *kernel* do Linux para os serviços centrais do sistema, tais como segurança, gestão de memória, gestão de processos, etc. O *kernel* também atua como uma camada de abstração entre o hardware e o resto do software.

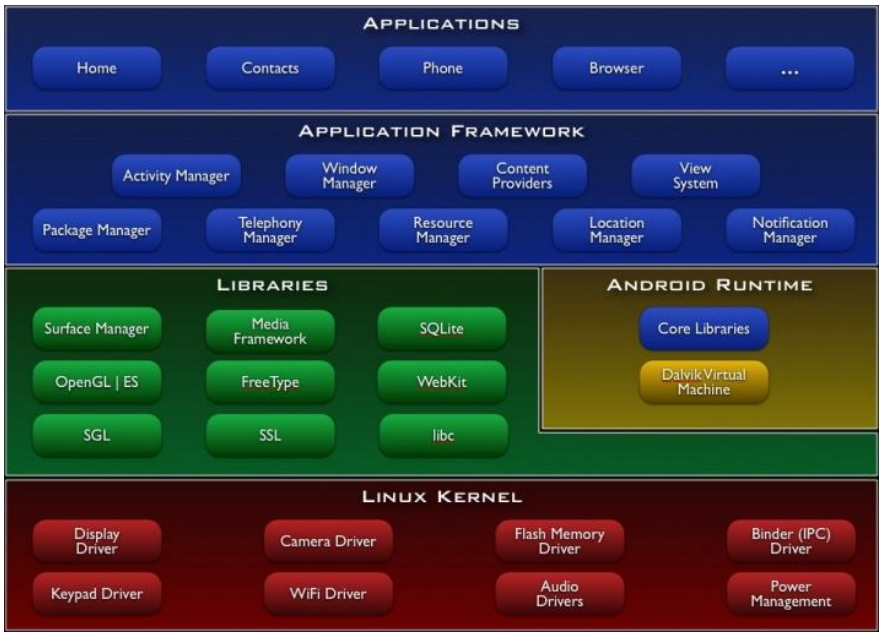


Figura 8.3. Arquitetura do Android [Google 2011].

Toda aplicação Android roda em seu próprio processo, com sua própria instância da máquina virtual Dalvik. O Dalvik não consome *byte-code* Java, mas sim *dexcode*. Para isso, o Google desenvolveu uma ferramenta, chamada "dx", que converte Java *bytecodes* (*.class) em *dexcodes* (*.dex).

Além disso, desde a versão 2.2 (Froyo), o Android possui uma implementação de JIT (*Just-in-time*), que compila *dexcodes* para a arquitetura-alvo em tempo de execução, tornando a execução dos processos consideravelmente mais rápidas, já que não precisa ficar interpretando *dexcodes*.

Como linguagem de programação base para desenvolvimento de suas aplicações, o Android utiliza o Java, além de fazer uso de um conjunto de bibliotecas C/C++ usadas por

diversos componentes expostas para os desenvolvedores através do *framework*. Abaixo estão algumas das principais bibliotecas disponíveis:

- *System* C *library* – uma implementação derivada da biblioteca C padrão sistema (libc) do BSD sintonizada para dispositivos rodando Linux.
- *Media Libraries* – baseado no PacketVideo's OpenCORE; as bibliotecas suportam os mais populares formatos de áudio e vídeo, bem como imagens estáticas.
- *Surface Manager* – gere o acesso ao subsistema de exibição bem como as múltiplas camadas de aplicações 2D e 3D.
- LibWebCore – um *web browser engine* utilizado tanto no Android *Browser* quanto para exibições *web*.
- SGL – o *engine* de gráficos 2D.
- 3D *libraries* – uma implementação baseada no OpenGL ES 1.0 APIs; as bibliotecas utilizam aceleração 3D via hardware (quando disponível) ou o software de renderização 3D altamente otimizado incluído no Android.
- FreeType – renderização de fontes *bitmap* e *vector*.

8.2.3. Configurando o Ambiente de Desenvolvimento

A fim de possuir todas as ferramentas necessárias para trabalhar no desenvolvimento de aplicativos para plataforma Android será necessário ter instalado os seguintes softwares:

- JDK (Java *Development* Kit)
- Android SDK
- Eclipse IDE
- ADT (Android *Development Tools*) *Plugin*

Essa seção tem por objetivo demonstrar como se dá a instalação desses softwares, além de configurações necessárias para iniciar a codificação para Android.

8.2.4. Instalando o Kit de Desenvolvimento Java (JDK)

O JDK possui as ferramentas necessárias para desenvolvimento de aplicações utilizando a linguagem Java. Pode ser baixado no seguinte link:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Na instalação do JDK a partir do arquivo baixado deve-se apenas seguir as instruções do programa instalador.

8.2.5. Instalando o Android SDK

Através do Android SDK pode-se criar as mais diversas aplicações para dispositivos que possuem o Android como Sistema Operacional. Pode ser baixado no seguinte link:
http://developer.Android.com/sdk/index.html

Na instalação do Android SDK a partir do arquivo baixado deve-se apenas seguir as instruções do programa instalador.

Caso seja exibida a mensagem “Java SE *Development* Kit (JDK) *not found*.” mesmo já tendo instalado o JDK, deve-se voltar e avançar novamente no instalador. A mensagem não se repetirá.

Utilizar o caminho de instalação: C:\Android\Android-sdk.

Ao término da instalação, rodar o SDK Manager, no qual serão identificadas e disponibilizadas atualizações das diversas Plataformas (versões) do sistema Android. A versão utilizada no minicurso será a Android 2.2 API 8, no entanto é aconselhável instalar todos os pacotes. Clicar em "*Accept All*" e em seguida, "*Install*".

8.2.6. Instalando o Eclipse IDE

O Eclipse é uma ferramenta de integração da linguagem de programação Java. Nele, também é necessário integrar o Android SDK para o desenvolvimento de aplicações Android. O Eclipse IDE pode ser baixado no link a seguir:
http://www.eclipse.org/downloads/

Dentre as opções apresentadas no *website* da IDE, será utilizada nesse minicurso o Eclipse IDE for Java *Developers*.

Para o Eclipse não é necessário instalação, pois os arquivos rodam direto da pasta onde foram descompactados, basta extrair o conteúdo do arquivo baixado para o local desejado.

8.2.7. Instalando o ADT Plugin

O ADT (Android *Development Tools*) é um *plug-in* para o Eclipse que foi projetado para fornecer um ambiente poderoso para desenvolvimento de aplicativos para Android. Seguem os passos para sua instalação:

1. Inicie o Eclipse, clique em *Help > Install New Software.*
2. Clique no botão *Add*.
3. Na janela que abrirá digite “*ADT Plugin*” para *Name* e a URL “*https://dl-ssl.google.com/Android/eclipse/*” para *Location.*
4. Após dar *OK*, na janela que segue, selecione *Developer Tools* e avance.
5. Será exibida a lista de ferramentas que serão baixadas. Avance novamente.
6. Aceite os termos de licença e clique em Finalizar.
7. Aguarde o término da instalação e reinicie o Eclipse.

8.2.8. Configurando o Android SDK e instanciando uma AVD

Estando instalado o ADT *Plugin* é necessário alterar as suas opções no Eclipse IDE para que aponte para a pasta de instalação do SDK Android. Seguem os passos para a configuração:

1. Acesse o caminho *Windows > Preferences > Android*.
2. Clique em *Browser*, localize a pasta de instalação da SDK Android (C:\Android\Android-sdk) e clique em *Ok*.

Pode-se executar uma aplicação desenvolvida através de um Dispositivo Virtual, que nada mais é que um emulador.

Para configurar um Dispositivo Virtual, abrir o *SDK Manager* (pode ser feito pelo Eclipse IDE acessando *Windows > Android SDK* e *AVD Manager*) e clicar em *new*.

Na tela em seguida, preencher os dados como segue na figura 8.4. Em seguida, clicando em *Create AVD*, um dispositivo é criado e está pronto para emular aplicações Android em seu computador. Selecione o dispositivo criado e clique em *Start* para inicializar o emulador.

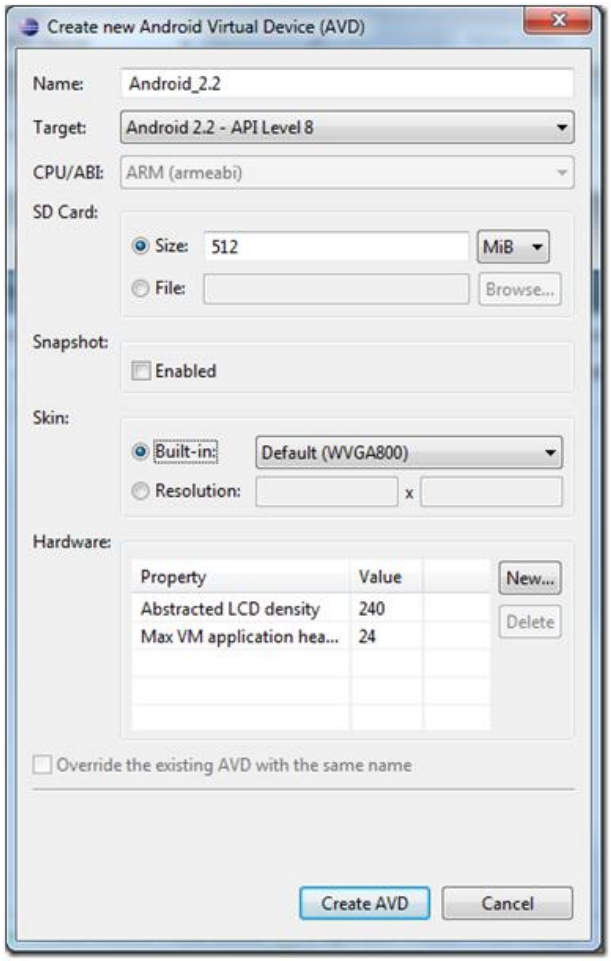


Figura 8.4. Configuração do AVD.

8.2.9. Criando um projeto utilizando Eclipse IDE

Nessa seção, será criado um projeto utilizando a IDE Eclipse. Além de simular a aplicação com o emulador, serão apresentadas os arquivos que compõem a estrutura básica de qualquer aplicação Android e as ferramentas de depuração.

Para criação de um novo projeto no Eclipse IDE, deve-se ir em *File > New > Project*. Ou clicando com o botão direito sob a área do *Project Explorer* e em seguida *New > Project*. Na tela que se abrirá selecione *Android Project* dentro da pasta Android e clique em *Next*.

Obs.: Caso não veja o Project Explorer, pode ir em *Windows* > *Show View* > *Project Explorer*.

Na tela seguinte, em *Project Name* digite um nome para seu projeto, nesse exemplo do minicurso utilizaremos o nome "MyApp". Deixe selecionada a opção C*reate New Project in Workspace* e *Use Default Location*. Então, pressione *Next*.

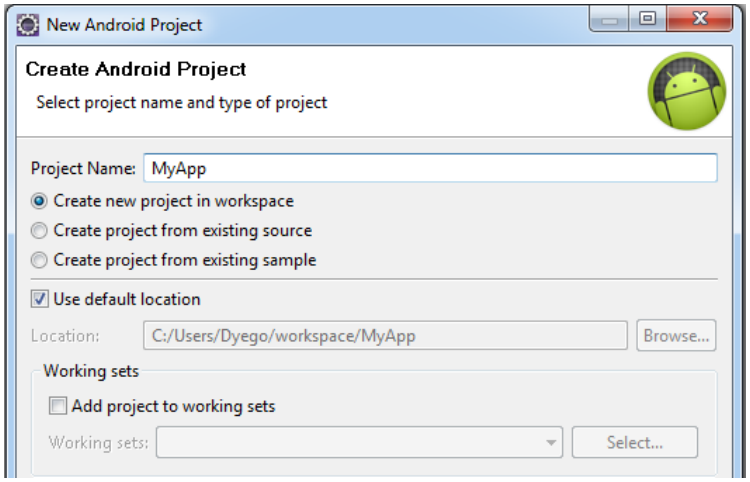


Figura 8.5. Declarando o nome da aplicação.

Na tela seguinte, escolhe-se a qual versão mínima do Android se vai desenvolver juntamente com sua API. No minicurso escolheremos a versão 2.2, API 8. Na tela posterior coloca-se o nome do pacote (Figura 8.6). Nesse exemplo do minicurso utilizaremos: "br.erbase.Android.myapp" e em seguida clica-se em *Finish*.

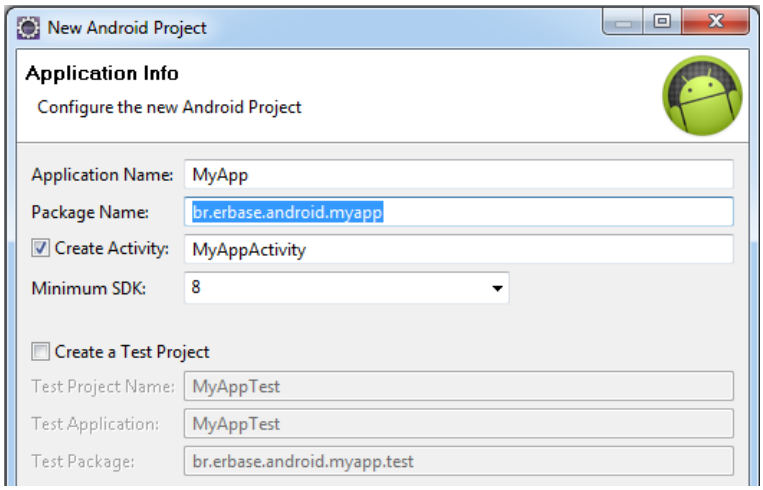


Figura 8.6. Definição do pacote principal da aplicação.

Feito isso, repare no *Project Explorer* que são gerados alguns arquivos por padrão. A próxima subseção explica os arquivos gerados.

8.2.10. Apresentação da estrutura de arquivos a partir do projeto criado

A estrutura básica de um projeto para Android é composta pelos seguintes elementos, ilustrados na figura 8.8:

- Pasta src: Inclui o pacote para a aplicação, que no exemplo é “br.erbase.Android.myapp”.

- Pasta gen: Inclui o pacote da aplicação com arquivos gerados.

- R.java: O ADT cria esse arquivo automaticamente, que representa as constantes necessárias para acessar diversos recursos da aplicação Android. Ou em outras palavras, esta classe é usada como um atalho para referenciar os recursos da pasta “res” no código.

- MyAppActivity.java: Implementação da classe principal da aplicação. Diferentemente das aplicações comuns de Java, toda classe para aplicação Android deve ser derivada da classe Activity (Atividade) e possui como método principal, o método onCreate. Dentro desse método ele invoca o método onCreate da super classe passando o mesmo parâmetro. Logo após esse método, vem o método setContentView, responsável por exibir a tela da minha aplicação, baseado nos *layouts* xml. Por padrão ele chama o arquivo “main.xml”. O código da Activity será apresentado mais detalhadamente na subseção 2.2.4.1.

- Bibliotecas de referência (Android <versão>): Contém o arquivo Android.jar, que é o arquivo da classe de *runtime* do Android, localizado no Android SDK.
- Pasta res/: Contém os recursos para a aplicação, incluindo:
    - Ícones (por exemplo, os arquivos ic_laucher.png separado em várias resoluções);
    - Arquivos de *layout* (o arquivo main.xml), nos quais são montados os elementos visíveis da interface com usuário das telas;
    - *Strings* (o arquivo strings.xml), que contém as constantes ligadas à *strings*.

- AndroidManifest.xml: Descritor de implementação da aplicação de exemplo. Contém uma descrição mais detalhada da aplicação do que a que é encontrada no arquivo main.xml. No exemplo, o código do *manifest* é o que segue:

MyApp Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.erbase.android.myapp"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MyAppActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

Figura 8.7. Arquivo *Manifest*.

De modo mais amplo, antes do sistema Android poder iniciar um componente de aplicação, o sistema deve saber que o componente existe lendo o arquivo

AndroidManifest.xml da aplicação (o arquivo *manifest*). Sua aplicação deve declarar todos seus componentes neste arquivo, que deve estar na raiz do diretório do projeto da aplicação. O *manifest* faz uma quantidade de tarefas em adição à declaração de componentes da aplicação, como:

- Identificar quaisquer permissões de usuário necessárias à aplicação, como acesso à Internet ou acesso de leitura nos contatos do usuário.
- Declarar o API Level mínimo necessário pela aplicação, baseado nas APIs que a aplicação usa.
- Declarar as características de hardware e software usadas ou requeridas pela aplicação, como câmera, serviço de Bluetooth, ou tela *multitouch*.
- Bibliotecas de API necessárias para a execução (além das nativas das APIs do framework Android), como a biblioteca do Google Maps.
- E muito mais.

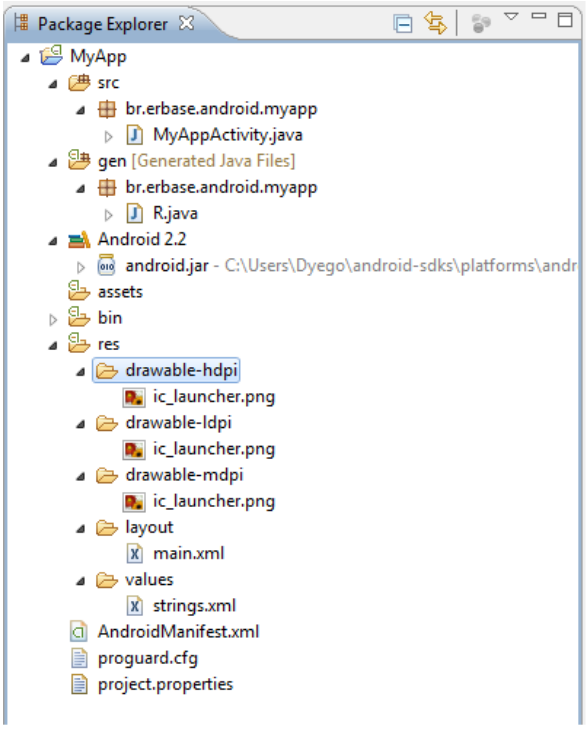


Figura 8.8. Elementos estruturais presentes em qualquer projeto Android.

8.2.11. Simulando a aplicação com o emulador

Com o intuito de testar sua primeira aplicação abaixo está o passo a passo para configurações básicas antes de simular a execução da aplicação no emulador.

Inicialmente, deve-se procurar o símbolo de *play* na barra de menu superior e em seguida, *Run Configurations*.

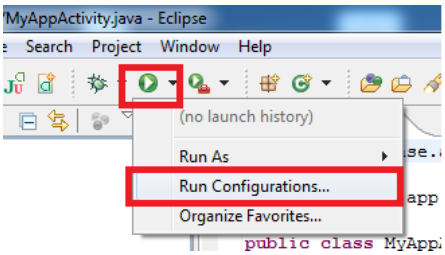


Figura 8.9. Opções de execução.

Na tela que se abrirá deve-se no lado esquerdo selecionar *New_configuration* (que está em *Android Application*) e no lado direito deve-se selecionar o projeto, no exemplo MyApp, ao clicar em *browser*. É possível também alterar o nome da configuração, nesse exemplo utilizou-se "MyApp_Sim".

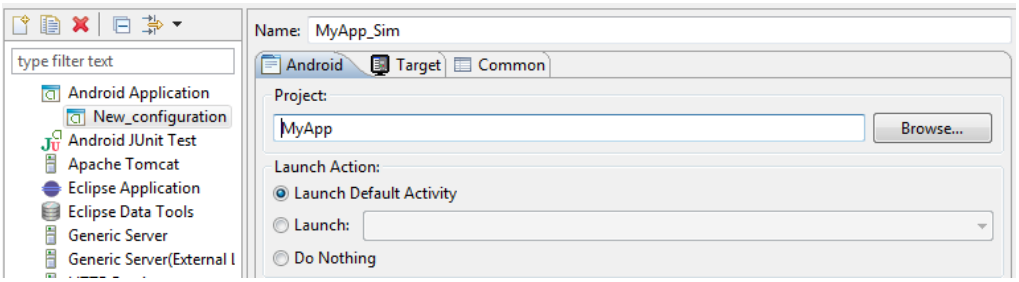


Figura 8.10. Painel Run Configurations.

Após isso, deve-se selecionar o emulador já cadastrado na aba *Target*. Em seguida, pode-se clicar em *Run*.

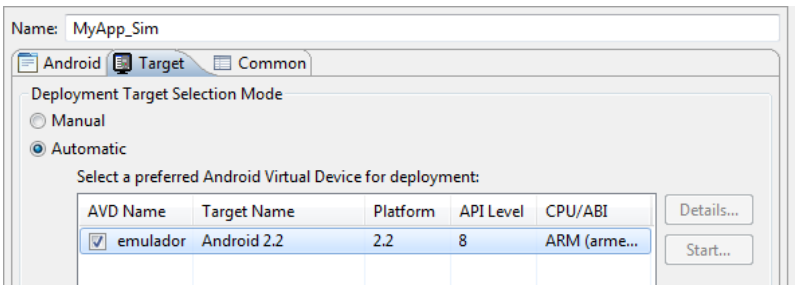


Figura 8.11. Seleção de Emulador.

Aguarde alguns minutos até que o emulador seja iniciado e, se necessário, destrave-o arrastando a barra esquerda para a direita, de modo semelhante a como se faz com o aparelho real. Então sua aplicação será instalada e executada.

Como o emulador simula um aparelho real, todas as aplicações contidas no *workspace* que foram executadas no mesmo AVD são carregadas quando o simulador é iniciado.

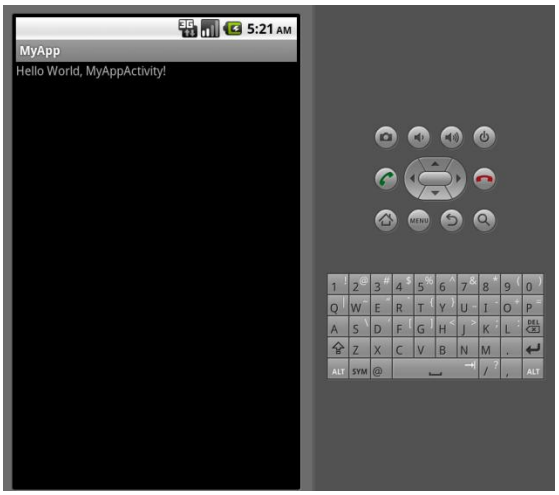


Figura 8.12. Aplicação rodando no Emulador.

8.2.12. Ferramentas de depuração

A plataforma Android possui algumas funcionalidades para auxiliar no trabalho de desenvolvimento de aplicativos. A seguir será mostrado o uso de duas ferramentas muito utilizadas no processo de simulação. Para utilizar essas ferramentas na IDE Eclipse o usuário deve navegar no menu *Window* até a opção *Show View*. Nesse menu, entre as opções mostradas se encontram o LogCat e o File Explorer que serão as ferramentas que irmos analisar.

Quando executamos uma aplicação escrita na linguagem Java o console do Eclipse exerce a função de mostrar as saídas do programas, que podem ser mensagem que o programador deseja que sejam exibidas ou até mesmo exceções que ocorreram durante a execução do código. No entanto, quando estamos utilizando o simulador o console se torna o responsável por mostrar informações relacionadas ao estado da simulação. Para observar informações sobre a execução do código foi criado o LogCat.

O LogCat traz informações sobre a execução do simulador e das aplicações. Caso ocorra uma exceção no código como uma exceção de ponteiro nulo, o erro aparecerá no LogCat. Essa ferramenta ajuda o desenvolvedor a achar erros de execução com maior facilidade.

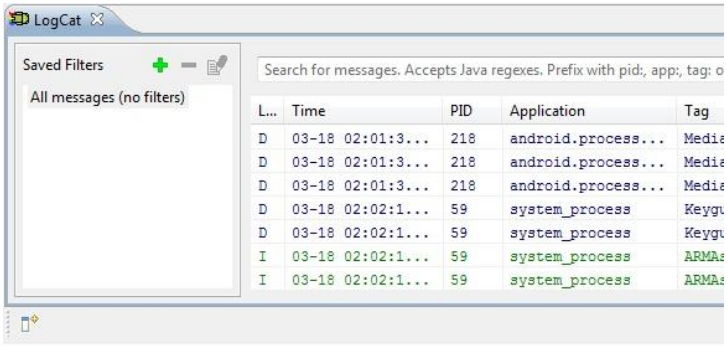


Figura 8.13. LogCat.

Já o File Explorer permite que o desenvolvedor tenha acesso ao esquema de pastas da simulação, podendo inclusive adicionar arquivos as pastas. Entre as pastas existentes, uma pasta que representa o cartão de memória do dispositivo chamada "sdcard". Poder inserir arquivos no simulador é um recurso útil, tendo em vista que aplicações que manipulam arquivos das mais diversas formas precisam ter um teste dessa manipulação antes de irem para o dispositivo de fato. Essa ferramenta é muito útil também para manipular arquivos de bancos de dados, extraindo esses arquivos e manipulando o banco com um gerenciador externo ao Eclipse IDE.

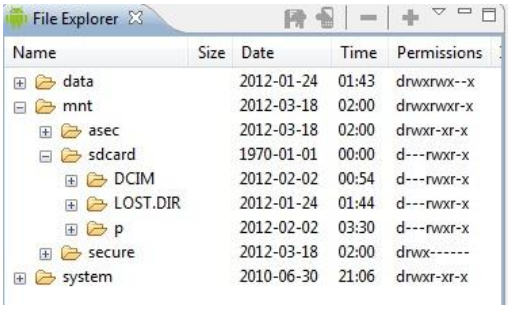


Figura 8.14. *File Explorer*.

8.2.13. Explorando a Plataforma

Uma aplicação Android possui blocos de construção essenciais. Cada bloco, ou componente, é um ponto diferente através do qual o sistema operacional pode entrar em sua aplicação. Nem todos os componentes são pontos de entrada reais para o usuário e alguns dependem uns dos outros, mas cada um existe como uma entidade própria e desempenha um papel específico, cada um é um bloco de construção singular que ajuda a definir o comportamento geral do aplicativo.

Há quatro tipos diferentes de componentes fundamentais de aplicação. Cada tipo tem uma finalidade distinta e tem um ciclo de vida distinta que define como o componente é criado e destruído. Abaixo estão os quatro tipos de componentes de aplicação:

- Activity

Uma Activity é, basicamente, uma classe gerenciadora de interface com o usuário. Quando uma aplicação Android é executada, na verdade é a sua Activity principal que é lançada. Por exemplo, uma aplicação de *e-mail* pode ter uma Activity que mostra a lista de novos e-mails, outra Activity para compor *e-mails*, e outra Activity para ler *e-mails*. Embora as Activities trabalhem juntas para formar uma experiência de usuário coesa na aplicação de *e-mail*, cada uma é independente das outras. Desta forma, uma aplicação diferente pode iniciar qualquer uma destas Activities (se a aplicação de *e-mail* permitir). Por exemplo, uma aplicação de câmera pode iniciar a Activity na aplicação de *e-mail* para compor um novo *e-mail*, visando o envio da foto por email. Uma Activity é implementada como uma subclasse da classe Activity. Falaremos mais dela em tópicos seguintes.

- Services

Um Service (serviço) é um componente que roda em *background* para executar operações demoradas ou para executar processos remotos. Um Service não possui interface do usuário. Por exemplo, um Service pode tocar música no *background*, enquanto o usuário está em uma aplicação diferente, ou pode carregar dados através da rede sem bloquear a interação do usuário com uma Activity. Outro componente, como uma Activity, pode iniciar o Service e deixá-lo rodando ou conectar-se nele para interagir com o mesmo. Um Service é implementado como uma subclasse de Service e devido ao foco do minicurso não serão trabalhados mais a fundo.

- Content Providers

Um Content Provider (provedor de conteúdo) gerencia um grupo de dados de aplicação compartilhados. Você pode armazenar dados no sistema de arquivos, uma base SQLite, na *web*, ou em qualquer outro local de armazenamento persistente que sua aplicação possa acessar. Através do Content Provider, outras aplicações podem consultar ou mesmo modificar os dados (se o Content Provider permiti-lo). Por exemplo, o sistema Android provê um Content Provider que gerencia as informações dos contatos do usuário. Desta forma, qualquer aplicação com as permissões apropriadas pode consultar parte do Content Provider (como um ContactsContract.Data) para ler e escrever informações sobre uma pessoa em particular.

Content Providers também são úteis para ler e escrever dados que estão privados na sua aplicação e não são compartilhados. Por exemplo, salvar notas pessoais.

Um Content Provider é implementado como uma subclasse de ContentProvider e deve implementar um conjunto de APIs que permite a outras aplicações executarem transações. Devido ao foco do minicurso não serão trabalhados mais a fundo.

- Broadcast Receivers

Um Broadcast Receiver (receptor de notificações) é um componente que responde a notificações a nível de sistema. Muitas notificações originam-se do sistema - por exemplo, uma notificação anunciando que a tela foi desligada, a bateria está fraca, ou uma foto foi tirada. Aplicações também podem iniciar *broadcasts* – por exemplo, para deixar outras aplicações sabendo que alguns dados foram baixados para o dispositivo e estão disponíveis para uso. Embora Broadcast Receivers não exibam interface ao usuário, eles podem criar notificações na barra de status para alertar o usuário quando um evento *broadcast* ocorre. Mais comumente, embora, um Broadcast Receiver é somente um "*gateway*" para outros componentes e é intencionalmente criado para fazer a menor quantidade de trabalho. Por exemplo, ele pode iniciar um Service para executar alguma tarefa baseada no evento.

Um Broadcast Receiver é implementado como uma subclasse de BroadcastReceiver e cada *broadcast* é entregue como um objeto Intent. Um aspecto único do design do sistema Android é que qualquer aplicação pode iniciar um componente de outra aplicação. Por exemplo, se você quer que o usuário tire uma foto com a câmera do

dispositivo, provavelmente existe uma outra aplicação que faz isso e que sua aplicação pode usar, ao invés de desenvolver uma Activity sua para tirar a foto. Você não precisa incorporar ou mesmo *linkar* código da aplicação da câmera. Ao invés disso, você simplesmente inicia a Activity na aplicação da câmera que tira a foto. Quando completada, a foto é retornado para sua aplicação e então você pode usá-la. Para o usuário, parece que a câmera é na verdade parte de sua aplicação.

Quando o sistema inicia um componente, ele inicia o processo para aquela aplicação (se ele ainda não está rodando) e instancia as classes necessárias para o componente. Por exemplo, se sua aplicação inicia a Activity na aplicação da câmera que captura a foto, a Activity roda no processo que pertence à aplicação da câmera, não no processo de sua aplicação. Entretanto, diferente da maioria dos outros sistemas, aplicações Android não possuem um único ponto de entrada (não existe uma função main(), por exemplo).

Devido ao sistema rodar cada aplicação em um processo separado com permissões de arquivo que restringem o acesso de outras aplicações, sua aplicação não pode ativar diretamente um componente de outra aplicação. O sistema Android, entretanto, pode. Então, para ativar um componente de outra aplicação, você deve entregar uma mensagem para o sistema que especifica sua intenção (Intent) de iniciar um componente em particular. O sistema então ativa o componente para você.

Nas subseções seguintes são abordados conceitos e aplicações relacionadas ao componente Activity. Adicionalmente, é apresentado o Intent, que exprime uma intenção de realização de algo da aplicação para o Sistema Operacional através de mensagens, e é necessário, dentre outras funcionalidades, para a navegação entre Activities.

8.3 Activity

Uma Activity é uma classe que herda de Android.app.Activity ou suas subclasses. Activity é responsável por tratar os eventos da tela como, por exemplo, tratar o clique do botão na tela, escrever um texto dinamicamente na tela, etc. As Activities sobrescrevem o método onCreate(Bundle) que é responsável por realizar a inicialização da tela através do método setContentView(view), essa *view* passada como parâmetro é a tela que será inicializada. Podemos interpretar Activity como tela.

No exemplo utilizado até aqui o código da Activity é o seguinte:

MyApp Manifest | *MyAppActivity.java

```
package br.erbase.android.myapp;

import android.app.Activity;
import android.os.Bundle;

public class MyAppActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Figura 8.15. Código da Activity.

8.3.1. Ciclo de vida de uma Activity

Um aplicativo normalmente consiste de múltiplas Activities que são normalmente ligadas umas as outras. Cada Activity pode começar outra Activity, a fim de executar ações diferentes. Cada vez que começa uma nova Activity, a Activity anterior é interrompida, mas o sistema preserva a atividade em uma pilha (a "pilha de volta"). Quando uma nova atividade começa, é empurrada para a pilha de volta e leva o foco do usuário. A pilha de volta usa LIFO (last in, first out) como mecanismo de fila, então, quando o usuário está em uma Activity e pressiona a tecla BACK, a atividade é removida da pilha (e destruída) e retoma a Activity anterior.

Quando uma atividade é parada por causa de uma nova Activity, há uma notificação da alteração no estado através de métodos de retorno do ciclo de vida da Activity. Existem vários métodos de retorno que uma Activity possa receber, devido a uma mudança em seu estado. Por exemplo, quando parado, uma Activity deve liberar todos os objetos grandes, como conexões de rede ou banco de dados. Quando a da Activity recomeça, você pode readquirir os recursos necessários e retomar as ações que foram interrompidas. Estas transições de estado são parte do ciclo de Activity.

Um Exemplo prático é quando você está jogando. A Activity que esta no topo da pilha é a Activity do jogo, as demais Activities que estão abaixo da pilha, podem estar em modo de pausa, totalmente parado ou executando em segundo plano. Mas se você está jogando e recebe uma ligação na hora do pênalti. O que acontecerá?

O Android colocará a aplicação de ligação no topo da pilha e a aplicação do jogo abaixo, temporariamente parada. Existem alguns métodos que controlam o ciclo de vida de uma aplicação. São eles, onCreate(), onStart(), onResume(), onPause(),onStop() e o onDestroy(). Também existem subníveis de um ciclo de vida, são eles, *entire lifetime*, *visibile lifetime* e o *foreground lifetime*. Abaixo está ilustrado o ciclo de vida da Activity, seguido de uma breve explicação sobre cada método de retorno.

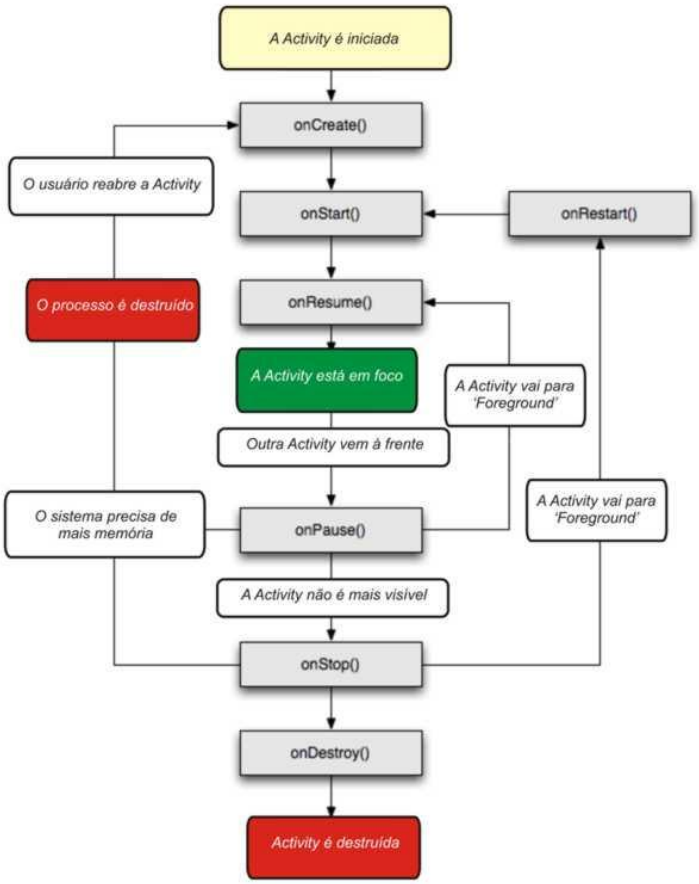


Figura 8.16. Ciclo de Vida da Activity.

- *onCreate()*: É a primeira função a ser executada quando uma Activity é lançada. Geralmente é a responsável por carregar os *layouts* XML e outras operações de inicialização. É executada somente uma vez durante a "vida útil" da Activity.

- *onStart()*: É chamada imediatamente após a onCreate() – e também quando uma Activity que estava em *background* volta a ter foco.

- o*nResume()*: Assim como a onStart(), é chamada na inicialização da Activity (logo após a própria onStart()) e também quando uma Activity volta a ter foco. Qual a diferença entre as duas? A onStart() só é chamada quando a Activity não estava mais visível na tela e volta a ter o foco, enquanto a onResume() sempre é chamada nas "retomadas de foco".

- *onPause()*: É a primeira função a ser invocada quando a Activity perde o foco (ou seja, uma outra Activity vem à frente).

- *onStop()*: Análoga à onPause(), só é chamada quando a Activity fica completamente encoberta por outra Activity (não é mais visível).

- *onDestroy()*: A última função a ser executada. Depois dela, a Activity é considerada "morta" – ou seja, não pode mais ser relançada. Se o usuário voltar a requisitar essa Activity, outro objeto será construído.

- *onRestart()*: Chamada imediatamente antes da onStart(), quando uma Activity volta a ter o foco depois de estar em *background*.

Dentre esses métodos os dois métodos de retorno mais importantes são:

- onCreate(): Deve-se implementar este método. O sistema chama-o ao criar a sua Activity. Dentro dle, você deve inicializar os componentes essenciais de sua Activity. Mais importante, este é o lugar onde se deve chamar setContentView() para definir o *layout* para a atividade do usuário a interface.

- onPause(): O sistema chama este método como o primeiro indício de que o usuário está saindo de sua Activity (embora nem sempre signifique que a atividade está sendo destruída). Nesse método geralmente é onde se deve fazer quaisquer alterações que devem ser mantidas para além da sessão atual do usuário (porque o usuário pode não voltar).

O diagrama apresentado na figura 8.16 é de fundamental importância para o correto entendimento do funcionamento de uma aplicação Android. Ele introduz, implicitamente, os estados que uma Activity pode estar, os quais estão explicados na figura 8.17, que segue:

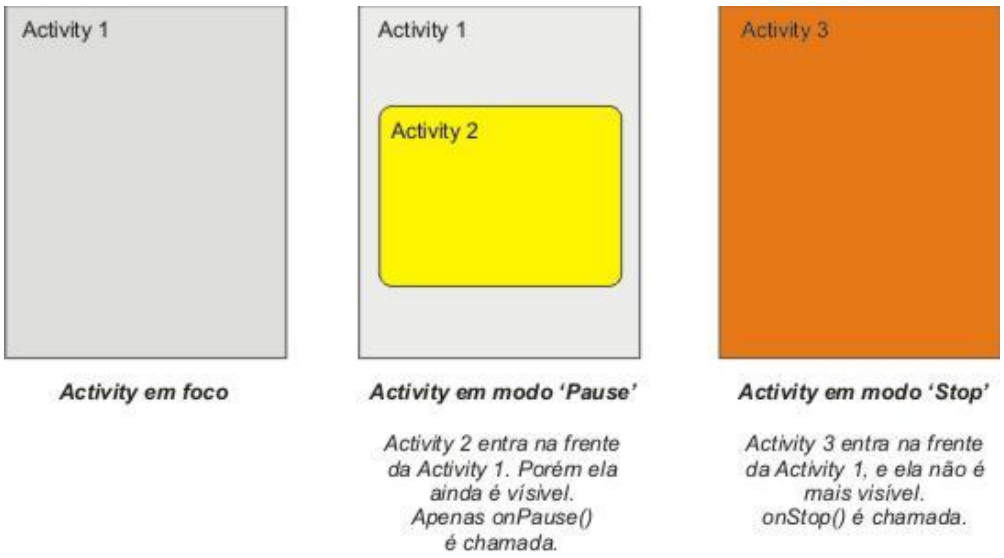


Figura 8.17. Exemplo de onPause() e onStop().

Uma atividade pode existir em três estados, essencialmente:

- Em foco: A atividade está em primeiro plano da tela e tem o foco do usuário.

- Em pausa: Outra atividade está em primeiro plano e tem foco, mas este é ainda visível. Ou seja, outra atividade é visível na parte superior de um presente e que a atividade é parcialmente transparente ou não cobre a tela

inteira. Uma atividade em pausa está completamente viva (o objeto Activity é mantido na memória, ele mantém todas as informações do estado e membro, e permanece preso ao gerenciador de janelas), mas pode ser morta pelo sistema em situações de pouca memória.

- Parado: A atividade é totalmente obscurecida por outra atividade (a atividade está agora em "*background*"). A atividade parada também está ainda viva (o objeto Activity é mantido na memória, ele mantém todas as informações do estado e membro, mas não está ligado ao gerenciador de janelas). No entanto, já não é visível para o usuário e pode ser morto pelo sistema quando a memória é necessária em outro lugar.

Se uma atividade está em pausa ou parada, o sistema pode retirá-la da memória, quer por pedir para terminar (chamando seu finish()), ou simplesmente matar o processo. Quando a atividade é aberta novamente (depois de ter sido concluído ou morto), ela deve ser criada por toda parte.

8.3.2. Inicialização de uma nova Activity e navegação entre telas

Já sabemos que quando uma aplicação é executada, a Activity definida como padrão (na criação do projeto) é lançada. Mas, obviamente, é possível criar outras Activities. E para executar outras Activities, basta usar a função startActivity(). No exemplo abaixo, lançamos uma segunda Activity a partir da principal.

Alterou-se o código da primeira Activity adicionando um botão, que ocupa toda a área da aplicação na tela, e implementando a interface OnCLickListener (figura 8.18). Além de passar uma String como parâmetro e inicializar uma nova Activity.

```
public class MyAppActivity extends Activity implements OnClickListener {
    static final int PICK_CONTACT_REQUEST = 0;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button b = new Button(this);
        b.setText("Clique aqui para abrir a tela.");
        b.setOnClickListener(this);

        setContentView(b);
    }

    public void onClick(View v) {
        Intent it = new Intent(this, Tela2.class);
        it.putExtra("msg", "Olá");
        startActivity(it);
    }
}
```

Figura 8.18. Alterações na Activity principal da aplicação.

Importante lembrar que toda nova Activity criada deve ser declarada no *Manifest*, sendo assim, este foi alterado também da seguinte forma:

```
<Activity Android:name=".Tela2" />
```

Criou-se uma nova classe que extende Activity com o nome de Tela2 que recebe a *String* passada pela Activity que a inicializou e exibindo-a no Log.

MyApp Manifest | MyAppActivity.java | Tela2.java

```
package br.erbase.android.myapp;

import android.app.Activity;

public class Tela2 extends Activity {

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        TextView view = new TextView(this);
        view.setText("Esta é a tela 2");
        setContentView(view);

        Intent it = getIntent();
        if(it != null){
            String msg = it.getStringExtra("msg");
            if (msg != null) {
                Log.i("ERBASE", "Mensagem: " + msg);
            }
        }
    }
}
```

Figura 8.19. Segunda Activity.

8.3.3. ListActivity

Quando é necessário listar dados em uma tela ou apresentar mapas existem tipos específicos (nesses casos, ListActivity e MapActivity, respectivamente) que encapsulam diversas características para facilitar ao programador. Por exemplo, a ListActivity já implementa o componente ListView. Nessa subseção será visto como usar uma ListActivity para mostrar ao usuário uma lista de dados. Para tal, foi adicionada ao projeto MyApp uma nova Activity chamada ListActivity1, na qual é criado um *array* de *String* e utilizado um *adapter* passando os dados do *array* para a lista da tela.

```
package br.erbase.android.myapp;

import android.app.ListActivity;

public class ListActivity1 extends ListActivity {
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        // Array de Strings para visualizar na Lista
        String[] itens = new String[] { "Nome 1", "Nome 2", "Nome 3" };

        // Utiliza o adaptador ArrayAdapter, para exibir o array de Strings na tela.
        ArrayAdapter<String> adaptador = new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, itens);

        setListAdapter(adaptador);
    }
    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        super.onListItemClick(l, v, position, id);

        // Pega o item naquela posição
        Object o = this.getListAdapter().getItem(position);

        String item = o.toString();

        //Exibe um alerta
        Toast.makeText(this, "Você selecionou: " + item, Toast.LENGTH_SHORT).show();
    }
}
```

Figura 8.20. Código de ListActivity1.

A ListActivity é uma classe que extende a classe Activity e possui o objetivo de mostrar ao usuário uma Lista (uma ListView). Em suma, é uma Activity com alguns métodos para gerenciamento de listas, criada com o intuito de facilitar a criação de telas com essa configuração. São muito comuns nas aplicações Android.

Importante lembrar de declarar a Activity no *Manifest* da forma que segue. IntentFilter com "action.MAIN" e "category.Launcher" informa ao Android para que essa Activity possa ser executada como principal e lançada para o topo imediatamente a inicialiação da aplicação.

```
<Activity Android:name=".ListActivity1">
      <Intent-filter>
        <action Android:name="Android.Intent.action.MAIN" />
        <category Android:name="Android.Intent.category.LAUNCHER" />
      </Intent-filter>
</Activity>
```

Para executar a aplicação com essa ListActivity1 como principal é necessário, antes de emular, ir em *Run* > *Configurations* e setar o campo *Launch* para "ListActivity1" (figura 8.21).

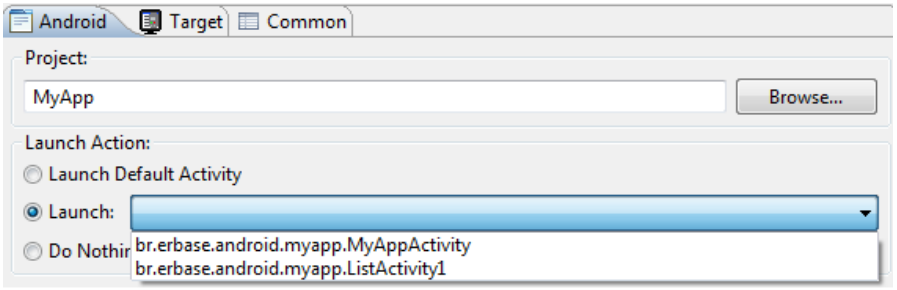


Figura 8.21. Setando ListActivity1 como principal.

Emulando, deve aparecer a seguinte tela (selecionando o terceiro item).

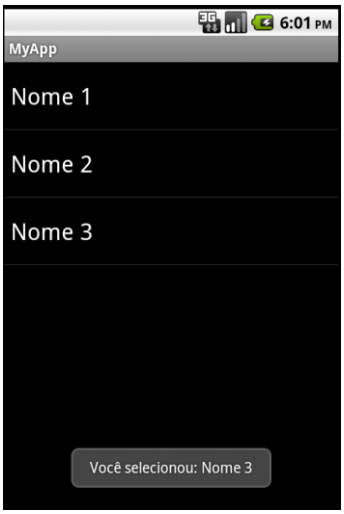


Figura 8.22. Emulando ListActivity.

8.3.4. Adapters

Adapters são classes responsáveis por fazer o que é chamado de "*bind*": Receber os dados de um Cursor (ou de alguma outra fonte de dados) e colocá-los nos seus respectivos lugares no *layout* da Activity.

Para Activities complexas, tipicamente são criadas subclasses da classe CursorAdapter (Adapter dedicado a tratar cursores). No exemplo utilizou-se apenas um ArrayAdapter que posuía um array de Strings. Ademais, utilizou-se um método para setar o ListActivity com esse array, chamado (setListAdapter).

8.3.5. Intent

Intent refere-se às intenções. São determinados comandos que se pode enviar ao Sistema Operacional Android para realizar alguma ação. Com Intents pode-se enviar ou recuperar dados. O método startActivity(Intent Intent) envia dados e requisições de telas, o método getIntent() recupera uma Intent enviada por meio do startActivity(), o método putExtra("nome_de_identificação", valor) insere na Intent algum valor, ou seja, o primeiro parâmetro é um nome para que possa ser identificado mais tarde e o segundo parâmetro um valor que pode ser String, boolean, int, float, etc. O getStringExtra("nome_de_identificação") pode recurar, por exemplo, uma String inserido na Intent através do putExtra("nome_de_identificação", valor).

8.3.6. Navegação entre telas e envio de parâmetros

Conforme mostrado no exemplo da subseção de Activities, é possível iniciar outra atividade, chamando startActivity(), passando-lhe uma Intent que descreve a atividade que deseja iniciar. A intenção especifica qualquer atividade exatamente o que deseja iniciar ou descreve o tipo de ação que deseja executar (e o sistema seleciona a atividade adequada para você, que pode mesmo ser de um aplicativo diferente). A intenção também pode transportar pequenas quantidades de dados a serem utilizados pela atividade que é iniciada.

Às vezes, pode-se querer receber um resultado da atividade que for inicializada a partir de outra. Nesse caso, deve-se iniciar a atividade chamando startActivityForResult() em vez de startActivity(). Para então receber o resultado da atividade subseqüente, aplicar o onActivityResult(), método de retorno. Quando a atividade subseqüente é feita, ele retorna um resultado de uma Intent para o seu método onActivityResult().

Por exemplo, talvez você queira que o usuário escolha um de seus contatos, para que sua atividade possa fazer algo com as informações desse contato. Pode-se ver como criar uma intenção e manipular o resultado com o código que segue:

```
private void pickContact() {
	URI Intent Intent = new Intent(Intent.ACTION_PICK, Contacts.CONTENT_URI);
	startActivityForResult(Intent, PICK_CONTACT_REQUEST);
}

@Override
```

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

       if (resultCode == Activity.RESULT_OK && requestCode ==
       PICK_CONTACT_REQUEST) {
              Cursor cursor = getContentResolver().query(data.getData(), new
       String[] { Contacts.DISPLAY_NAME}, null, null, null);
              if (cursor.moveToFirst()) {
                     Index = cursor.getColumnIndex(Contacts.DISPLAY_NAME);
                     String name = cursor.getString(columnIndex);
              }
       }
}
```

Este exemplo mostra a lógica básica que se deve usar no método onActivityResult() para lidar com um resultado de uma Activity. A primeira condição verifica se a solicitação foi bem-sucedida, se for, então o resultCode será RESULT_OK e se a solicitação para que este resultado está respondendo é conhecido, neste caso, o requestCode coincide com o segundo parâmetro enviada com startActivityForResult(). De lá, o código manipula o resultado de atividade, consultando os dados retornados de uma Intent.

Quando a segunda Activity terminar a sua execução, a função onActivityResult() será invocada, com o "resultado" como parâmetro. Mas como uma Activity define o seu resultado, a ser lido por aquela que a chamou ao invocar-se a função setResult (int resultCode), como por exemplo:

```
setResult(Intent.RESULT_OK);
```

8.4 Componentes de Interface Gráfica

A plataforma Android fornece uma série de componentes de interface gráfica desenvolvidos para atender as necessidades de aplicações embarcadas. Para posicionar e modificar a forma desses componentes a plataforma se utilizou de um esquema onde os componentes são declarados em um arquivo XML. A partir do momento que esses elementos são adicionados eles recebem um identificador único e são tratados pelo sistema como um recurso. Observe o exemplo de um arquivo de *layout*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Digite seu nome"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <requestFocus />
    </EditText>


    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Click aqui!" />

</LinearLayout>
```

Figura 8.23. Exemplo de *layout* em XML.

Apesar da criação desse arquivo de *layout* ser uma tarefa simples. Existem ferramentas que permitem o desenho dessas telas no modo gráfico, o que facilita na criação de novas aplicações e manutenção de aplicações existentes. Observe a imagem abaixo onde é apresentada a interface de desenho de interface do Eclipse. Nessa interface os componentes são arrastados para tela, sem que haja a necessidade de escrever o arquivo de *layout*.

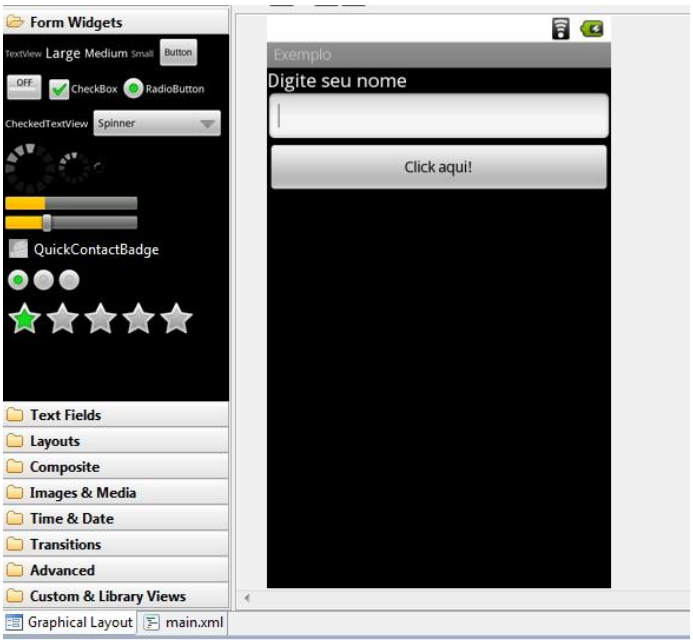


Figura 8.24. Tela de desenho de interfaces do Eclipse.

Uma vez montado o *layout* o código da aplicação pode fazer uso dos componentes que foram adicionados. Para manipular os componentes como objetos é preciso declarar uma variável do tipo do componente e fazer a ligação entre essa variável e o componente. Para estabelecer essa ligação é utilizado o método findViewById(id) herdado da classe Activity. O método nos retorna um objeto do tipo *View* e por isso há a necessidade de fazer uma operação de conversão de tipo antes de atribuir seu retorno a variável.

No exemplo abaixo a variável "edit" do tipo "EditText" recebe um valor vindo do método de ligação. Depois que a ligação é estabelecida as mudanças deitas na variável mudaram a forma e comportamento do componente. No exemplo o texto do componente está sendo alterado pelo método setText().

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.EditText;

public class ExemploActivity extends Activity {
    /** Called when the activity is first created. */

    EditText edit;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        edit = (EditText) findViewById(R.id.editText1);
        edit.setText("seu nome");

    }
}
```

Figura 8.25. Código de exemplo findViewById().

A seguir serão apresentados alguns componentes gráficos disponíveis na plataforma. O objetivo dessa seção é mostrar as características de cada componente e demonstrar seu uso com exemplos.

8.4.1. TextView

Esse componente é responsável por mostrar textos que não podem ser editado pelo usuário final. Esse componente é comumente usado para fazer rótulos de outros componentes (*label*) e inserir textos que devem apenas ser lidos pelo usuário final.

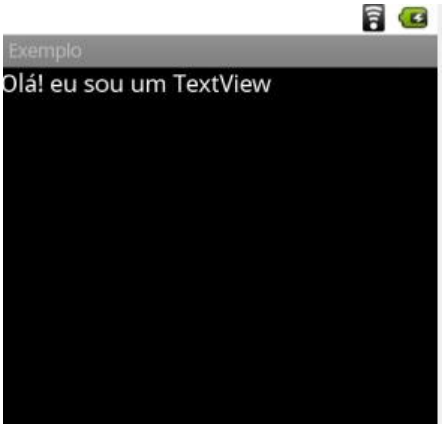


Figura 8.26. Exibição de TextView.

Para inserir um TextView na aplicação é declarada a *tag* TextView no arquivo de *layout* como no exemplo a seguir.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Olá! eu sou um TextView"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>
```

Figura 8.27. XML do TextView.

Todos os componentes possuem suas *tag*s de inserção no arquivo de *layout*. Para diminuir a repetição de código, nos próximos componentes serão mostradas apenas suas *tag*s.

8.4.2. EditText

Esse componente serve para o usuário inserir textos, sejam esses de uma ou múltiplas linhas, numéricos ou não. A configuração dessas peculiaridades (mascaras, tamanho de texto e outras propriedades) pode ser inserida nos atributos da *tag* que é declarada no arquivo de *layout*.

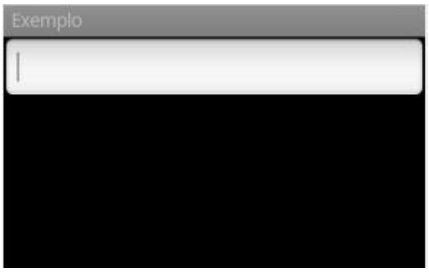


Figura 8.28. Exibição do EditText.

Para adicionar esse componente é utilizada a *tag* EditText. No exemplo a seguir dois componentes desse tipo são mostrados. Note que um deles possui uma máscara numérica, o que impede que caracteres que não sejam numéricos sejam adicionados.

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <requestFocus />
</EditText>

<EditText
    android:id="@+id/editText2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal" />
```

Figura 8.29. XML do EditText.

8.4.3. Alert

Esse elemento serve para mostrar uma mensagem de alerta para o usuário. A mensagem de alerta se sobrepõe a aplicação, só sendo possível continuar a execução quando a janela de alerta concluir sua tarefa.

Figura 8.30. Exibção de Alert.

Esse componente não é declarado no arquivo de *layout*. Para utilizá-lo é necessário fazer uma chamada no código como no exemplo a seguir.

```
AlertDialog.Builder dialogo = new AlertDialog.Builder(ExemploActivity.this);
            dialogo.setTitle("Resultado");
            dialogo.setMessage("Teste alerta");
            dialogo.setNeutralButton("Ok",null);
            dialogo.show();
```

Existe ainda a possibilidade de inserir mais de um botão no Alert, podendo esse ser usado como um elemento de confirmação ou decisão da próxima ação a ser tomada. Observe o exemplo abaixo onde ao clicar no botão “SIM” outro Alert é chamado

```
final AlertDialog.Builder dialogo = new Builder(ListaTelefonica.this);
dialogo.setTitle("Aviso");
dialogo.setMessage("Tem ceteza que deseja remover o contato?");
dialogo.setPositiveButton("Sim", new DialogInterface.OnClickListener() {

    public void onClick(DialogInterface dialog, int which) {
        // TODO Auto-generated method stub

        contatos.remove(arg2);
        lv.setAdapter(aContatos);
        final AlertDialog.Builder dialogo2 = new Builder(ListaTelefonica.this);
        dialogo2.setTitle("Aviso");
        dialogo2.setMessage("Removido com sucesso");
        dialogo2.setNeutralButton("OK",null);
        dialogo2.show();
    }
});
dialogo.setNeutralButton("Não",null);
dialogo.show();
```

Figura 8.31. Código de AlertDialog em Activity.

8.4.4. Toast

O componente Toast é utilizado para mostrar mensagens ao usuário que não precisam de confirmação. Esse componente é geralmente utilizado para informar ao usuário que alguma tarefa foi concluída.

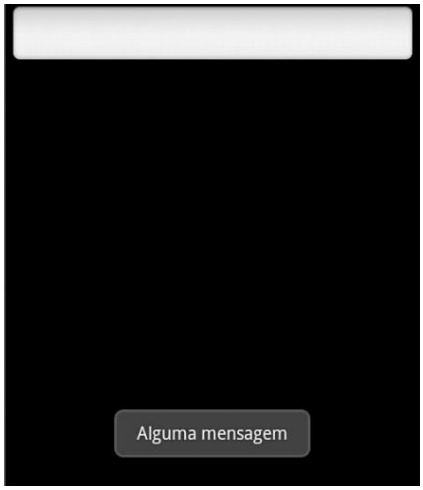


Figura 8.32. Exibição de Toast.

Como o Alert esse componente não é declarado no arquivo de *layout*, pois sua chamada é geralmente relacionada a um evento. A seguir observe um exemplo do código necessário para montar o Toast.

```
Toast.makeText(getBaseContext(),"Alguma mensagem", Toast.LENGTH_SHORT).show();
```

8.4.5. Button

O Button é o componente que representa um botão. Esse componente geralmente é utilizado para disparar eventos quando o mesmo é clicado. Botões são componentes muito utilizados em aplicações tradicionais. A eles são delegadas tarefas como mudar de trela e execução de ações

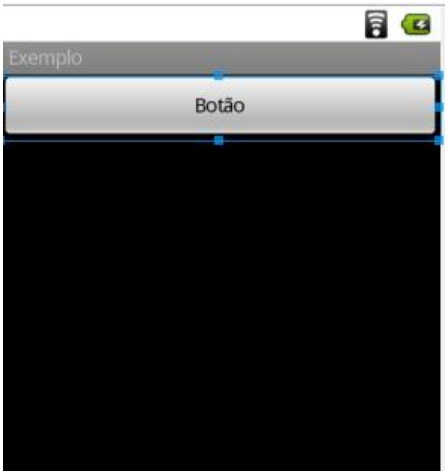


Figura 8.33. Exibição de Button.

Para inserir esse elemento no *layout* é utilizado a *tag* Button, que possui uma série de propriedades para alterar sua forma.

```
<Button
        Android:id="@+id/button1"
        Android:layout_width="match_parent"
        Android:layout_height="wrap_content"
        Android:text="Botão" />
```

Para vincular eventos ao clique do botão utilizamos o método "setOnClickListener()" no qual declaramos o objeto que será responsável por "ouvir" quando esse evento ocorrer.Observe o exemplo a seguir onde ao clicarmos no botão o evento de clique mostra um Alert na tela informando ao usuário que o evento ocorreu.

```
public class ExemploActivity extends Activity {
    Button bt;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        bt = (Button) findViewById(R.id.button1);

        bt.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            AlertDialog.Builder dialogo =new Builder(ExemploActivity.this);
            dialogo.setTitle("Click");
            dialogo.setMessage("O botão foi clicado");
            dialogo.setNeutralButton("Ok",null);
            dialogo.show();
            }
        }
    );}
}
```

Figura 8.34. Código de tratamento de evento de Button em uma Activity.

8.4.6. RadioButton e RadioGroup

RadioButton é um componente que geralmente é usado em grupo, de forma que apenas um desses itens que pertençam ao mesmo grupo pode estar marcado por vez. Para definir um grupo de RadioButtons é utilizado o componente RadioGroup.

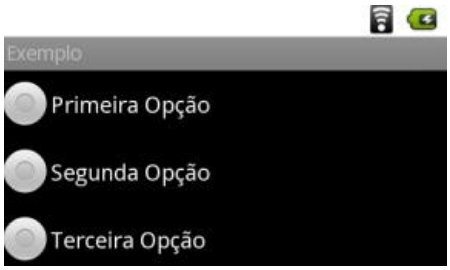


Figura 8.35. Exibição de RadioGroup.

Para que os RadioButtons façam parte do mesmo RadioGroup é preciso que no arquivo de *layout* as *tags* "<RadioButton>" estejam dentro de uma única *tag* "<RadioGroup>" como no exemplo abaixo:

```
<RadioGroup
        Android:id="@+id/radioGroup1"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content" >
        <RadioButton
            Android:id="@+id/radio0"
            Android:layout_width="wrap_content"
            Android:layout_height="wrap_content"
            Android:checked="true"
            Android:text="RadioButton" />
        <RadioButton
            Android:id="@+id/radio1"
            Android:layout_width="wrap_content"
            Android:layout_height="wrap_content"
            Android:text="RadioButton" />
        <RadioButton
            Android:id="@+id/radio2"
            Android:layout_width="wrap_content"
            Android:layout_height="wrap_content"
            Android:text="RadioButton" />
    </RadioGroup>
```

8.4.7. CheckBox

Checkbox é um elemento que possui dois estados, marcado ou desmarcado. A diferença do Checkbox em relação ao RadioButton é que as ações que ocorrem em um elemento Checkbox não interfere no estado de outros Checkbox. O elemento Checkbox é geralmente utilizado quando em um grupo de opções mais de uma pode ser selecionada.

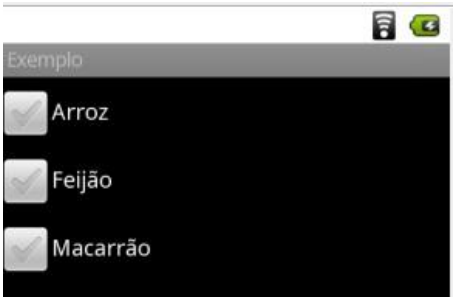


Figura 8.36. Exibição de CheckBox.

No arquivo de *layout* o elemento Checkbox é inserido com a *tag* "<CheckBox>" como no exemplo a seguir:

```
<CheckBox
        Android:id="@+id/checkBox1"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="Arroz" />
<CheckBox
        Android:id="@+id/checkBox2"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="Feijão" />
<CheckBox
        Android:id="@+id/checkBox3"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="Macarrão" />
```

Para manipular um objeto do tipo Checkbox através do código utilizamos a classe Checkbox. No exemplo a seguir é mostrado um exemplo de aplicação onde ao mudar o estado do Checkbox (marcado ou desmarcado) é exibido um Toast informando do estado atual.

```
public class ExemploActivity extends Activity {
    /** Called when the activity is first created. */

CheckBox ch;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
       ch = (CheckBox) findViewById(R.id.checkBox1);

        ch.setOnCheckedChangeListener(new OnCheckedChangeListener() {

            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                String s;
                if(isChecked){
                    s = "Marcado";

                }
                else{
                    s = "Desmarcado";

                }
                Toast.makeText(getBaseContext(),s,Toast.LENGTH_SHORT).show();
```

Figura 8.37. Código de tratamento de CheckBox em Activity.

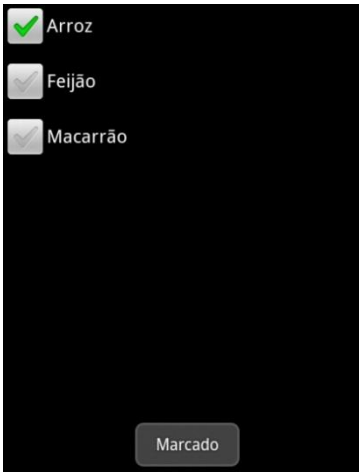


Figura 8.38. Exibição de CheckBox gerado a partir da Activity da figura anterior.

8.4.8. ImageView

ImageView é o componente responsável por mostrar imagens na tela. Essas imagens podem estar entre os arquivos do projeto, por fazer parte do *layout* da aplicação ou pode estar numa pasta externa e ser incluída em tempo de execução no componente.

Figura 8.39. Exibição de ImageView.

Para adicionar uma ImageView no projeto é utilizada a *tag* "ImageView". Essa *tag* possui como atributo obrigatório o "src" que é o caminho do arquivo de imagem que será exibido. Esse atributo, apesar de ser obrigatório pode ter valor vazio e assim não exibirá nenhuma imagem. Essa possibilidade de não passar valor nenhum é muito útil tendo em vista que essa imagem pode vim de fora da aplicação (de uma pasta ou até mesmo da internet). Observe o exemplo da declaração da imagem no arquivo de *layout*.

```
<ImageView
   Android:id="@+id/imageView1"
   Android:layout_width="178dp"
   Android:layout_height="270dp"
   Android:src="@drawable/ic_launcher" />
```

8.4.9. Componentes que utilizam ArrayAdapter

Alguns componentes comumente usados em formulários (como listas e *combobox*) possuem opções de valores pré-definidos. Para armazenar essas opções a plataforma Android possui a classe ArrayAdapter. A utilização dessa classe permite que itens sejam inseridos e removidos em tempo de execução, tornando a manipulação desse tipo de elemento uma tarefa prática.

A seguir serão abordados o uso de componentes que utilizam a classe ArrayAdapter para armazenar elementos. O uso dessa classe será melhor explicado no exemplo de uso de cada componente.

8.4.10. Spinner

O componente Spinner é conhecido também como c*ombobox* ou *dropdown* em outras linguagens. Esse componente permite que o usuário escolha entre as opções que ele oferece. Ao clicar numa opção essa é mostrada no componente, mas caso o usuário deseje mudar de opção ele deve clicar no componente para que as demais opções apareçam na tela.

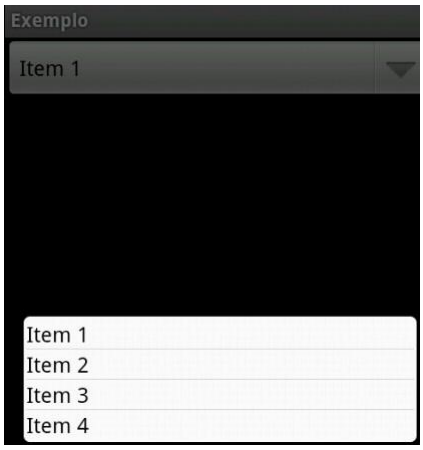


Figura 8.40. Exibição de Spinner.

A *tag* necessária para declarar um componente Spinner no arquivo de *layout* é a *tag* "Spinner", como mostrado no exemplo abaixo

```
<Spinner Android:id="@+id/spinner1"
     Android:layout_width="match_parent"
     Android:layout_height="wrap_content" />
```

Para modificar os elementos no componente utilizamos o método setAdapter(). Dessa forma, toda operação que precisa ser feita nos itens não são feitas no componente, pois basta modificar o adaptador que os novos dados são mostrados. Observe o exemplo a seguir onde uma operação de inserção de itens em um Spinner é feita

```
public class ExemploActivity extends Activity {
    Spinner sp;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        String itens[] = {"Item 1","Item 2","Item 3","Item 4"};
        sp = (Spinner) findViewById(R.id.spinner1);

        ArrayAdapter<String> a = new ArrayAdapter<String>(ExemploActivity.this,
                                    android.R.layout.simple_spinner_item,itens);
        sp.setAdapter(a);
    }
}
```

Figura 8.41. Código exemplo de inserção de itens em Spinner.

8.4.11. ListView

ListView é um componente que mostra um ou mais uma lista. Assim como o Spinner, esse componente pode possuir vários valores pré-estabelecidos.

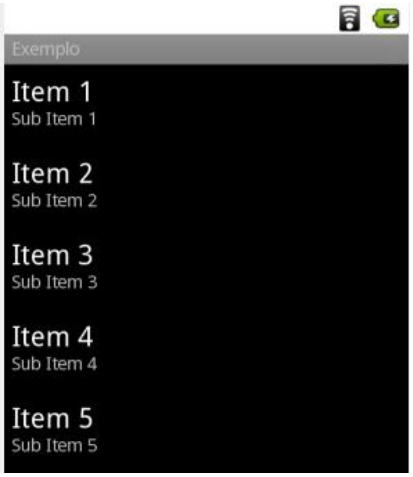


Figura 8.42. Exibição de ListView.

Para declarar uma ListView é utilizada a *tag* "ListView" no arquivo de *layout* como mostrado no exemplo abaixo:

```
<ListView Android:id="@+id/listView1"
 Android:layout_width="match_parent"
 Android:layout_height="wrap_content" >
 </ListView>
```

8.5 Instalação em dispositivo real

Uma das formas de instalar um aplicativo seu em um dispositivo real é: (i) gerando o .apk, (ii) passando para seu dispositivo via USB e (iii) instalando-o no aparelho. A seguir serão mostrados passos para geração do .apk de um exemplo de aplicação CRUD.

1. Para gerar o arquivo instalador .apk, deve-se, no *Project Explorer* do Eclipse IDE, clicar com o botão direito do mouse sob o título do projeto e selecionar a opção *Export*.

2. Na tela que se abrirá deve-se selecionar *Export Android Application*, em Android e clicar em *Next*.

3. Na tela seguinte, irá mostrar o título do projeto selecionado, clique apenas em *Next*.

4. Na tela seguinte selecione *Create New Keystore* e escolha um local para salvar o *keystore* dando um nome a ele, além de configurar uma senha com pelo menos 6 caracteres.

5. Na tela posterior, configure como segue e clique em next:

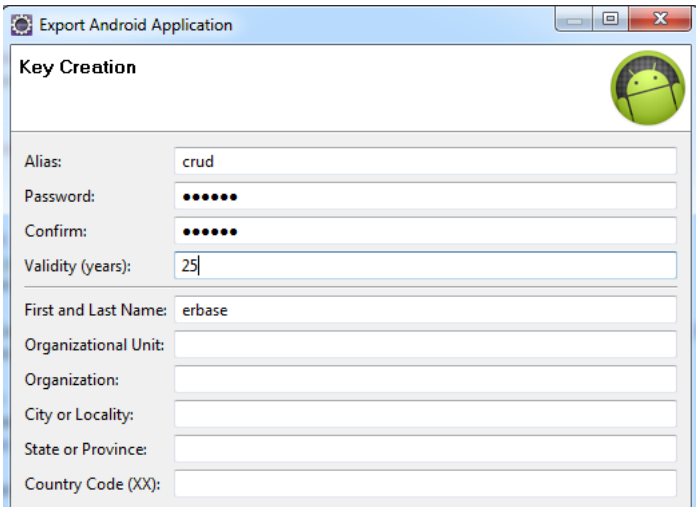


Figura 8.43. Criação de Keystore.

6. Em seguida escolha um local para salvar o arquivo .apk dando um nome a ele e clique em *Finish*.

Para instalar o .apk gerado no seu dispositivo móvel real, siga os passos a seguir.

1- Habilite no *smartphone* ou *tablet* (em *configurações* > *aplicativos*) a opção permitir instalar aplicativos desconhecidos.

2- Conecte o seu *smartphone* ou *tablet* ao computador.

3- Ao conectar será necessário escolher a opção de conexão USB, deve-se escolher a opção armazenamento de dados.

7. Obs.: Esta escolha pode não ser solicitada, nestes casos deve-se ir na barra de notificações e clicar em conexão USB e depois escolher a opção armazenamento de dados.

4- Copie o blackbox.apk para seu *smartphone* ou *tablet*.

5- Desconecte o USB.

6- Execute o arquivo "blackbox.apk" no seu aparelho.

7- A instalação prosseguirá. Dê permissão para acesso a Internet.

8.6 Bibliografia utilizada

Anatel (2012) Consolidação Serviços Móveis no Brasil,
Disponível em: <http://sistemas.anatel.gov.br/stel_java/pdf.do?comando=doPdf>. Acesso em Fevereiro de 2012.

Android (2010) Android. Disponível em: <http://www.Android.com/>. Acesso em Fevereiro de 2012.

Apple (2011). iPhone. http://www.apple.com/iphone/.

B'Far, R. (2004). Mobile Computing Principles: Designing and Developing Mobile Applications with UML and XML, Cambridge University Press.

BlackBarry (2004), www.blackberry.com.

Google. Android Developers. 2011. Disponível em: <http://developer.Android.com>. Acesso em: 10 de dezembro de 2011.

Lecheta, R. R. Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK. 2ª Edição Revisada e Ampliada. São Paulo: Novatec Editora, 2010.

Lee, Valentino et al. (2004). Mobile Applications: Architecture, Design, and Development, Prentice Hall

Nokia (2010) Symbian. http://symbian.nokia.com/.

Portal Android - Comunidade de Desenvolvedores da Plataforma Android. 2011. Disponível em: <http://www.portalAndroid.org/comunidade/>. Acesso em: 11 de dezembro de 2011.

StatCounter. 2012. Disponível em: <http://gs.statcounter.com/#mobile_os-ww-monthly-201201-201203-bar>. Acesso em Fevereiro de 2012.

Silveira, F. 2011. Curso Desenvolvendo para Android. Disponível em: <http://www.felipesilveira.com.br/desenvolvendo-para-Android/>. Acesso em Fevereiro de 2012.

Capítulo 9

Análise de Vulnerabilidades em Mecanismos de Segurança em Redes Wi-Fi

Rafael Souza, Kádna Camboim e Jean Araujo

Abstract

This chapter analyzes the main characteristics of wireless networks, norms and protocols, as well as security issues arising from its structure. Some filtering techniques to identify the SSID, MAC address filtering and encryption techniques such as WEP, TKIP, WPA and WPA2 are also defined. Techniques for capturing information as Wardriving and Warchalking are respected, and some types of intrusion attacks that are able to circumvent security mechanisms.

Resumo

Neste capítulo serão apresentadas as principais características das redes sem fio, suas normas e protocolos, bem como problemas de segurança decorrentes de sua estrutura. Algumas técnicas de filtragem como identificação pelo SSID, filtragem por endereço MAC e técnicas de encriptação como WEP, TKIP, WPA e WPA2 também serão definidas. Técnicas para captura de informações como Wardriving e Warchalking serão conceituadas, além de alguns tipos de ataques de intruso que são capazes de burlar mecanismos de segurança.

9.1. Introdução

A crescente evolução tecnológica mundial vem influenciando diversos setores do mundo, principalmente os de telecomunicações. Com a difusão da Internet e o elevado aumento no número de usuários, as diferentes formas de interconexão aparecem como alternativas para interligar pessoas separadas geograficamente. Internet é uma rede mundial de computadores capaz de conectar milhões de equipamentos em todo o mundo [Kurose e Ross 2006].

No decorrer das últimas décadas, as redes de computadores se tornaram bastante utilizadas por públicos de todas as idades, ganhando o mérito de grande difusora de informações. Foram desenvolvidas tecnologias para permitir acesso e comunicação entre

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

computadores de um ponto a outro. Essa comunicação entre dois ou mais computadores é chamada de rede de computadores.

Rede de computadores é onde duas ou mais máquinas autônomas interconectadas se comunicam e trocam informações por meio de uma tecnologia, cujos meios de transmissão podem estar confinados ou não [Tanenbaum 2011]. Os meios de transmissão de informações entre computadores passaram por avanços tecnológicos no decorrer de sua evolução. Isso aconteceu devido à necessidade de melhor estrutura existente.

As redes de computadores surgiram com a necessidade de trocar informações, acessar dados e compartilhar recursos, o que diminui o trabalho de armazenar arquivos em mídia e custos com compras de hardware [Torres 2001].

Segundo [Albuquerque 1999] as redes tornaram possível a implementação dos sistemas distribuídos de computação que as utilizam para trocar dados e informações. Uma das formas básicas de classificação de uma rede está relacionada com a extensão física, assim temos:

- Rede de Longa Distância (*Wide Area Network - WAN*) – rede geograficamente distribuída com uma grande área de abrangência, como um país ou continente;
- Rede Metropolitana (*Metropolitan Area Network - MANs*) – chegam a atingir alguns quilômetros de extensão, abrangendo cidades;
- Rede Local (*Local Area Network - LAN*) – redes privadas contidas em um único edifício ou campus universitário com até alguns quilômetros de extensão.

Cada tipo de rede apresenta velocidades de transmissão, topologias e formas de acesso ao meio que as diferenciam e caracterizam. Para possibilitar a comunicação entre os computadores são utilizados diferentes protocolos, a exemplo do IP, encarregado de transportar os dados que trafegam na internet, TCP que estabelece a conexão, oferece serviço de confirmação de entrega e retransmissão de pacotes, UDP encarregado de entregar mensagens, dentre outros [Tanenbaum 2011].

Durante muitos anos, mais precisamente até o final dos anos 90, a transmissão de dados em redes de computadores era realizada somente através de cabos, mas a partir do final dessa mesma década, outra tecnologia foi difundida através de conexões sem fio. Essa tecnologia de rede vem tomando o espaço das redes locais cabeadas e cada vez mais os usuários migram para esse tipo de estrutura. A razão dessa mudança está na simplicidade e mobilidade, que vêm ganhando o mercado dia após dia.

De acordo com a organização [JiWire 2012], o Brasil está entre os 20 países do ranking que possuem os serviços. Dentre as cidades, São Paulo é a nona na classificação mundial. As redes receberam novas padronizações como as LANs 802 e uma série de normas para as tecnologias por ela utilizadas, dentre as quais está a 802.11 *Wireless* LAN Working Group.

9.2. Redes sem fio – 802.11

A ideia da comunicação sem fio surgiu há mais de um século, com o padre brasileiro Roberto Landell de Moura, quando o mesmo realizava as primeiras transmissões radiofônicas com aparelhos transmissores e receptores de sua invenção. Um ano depois, o físico italiano Guglielmo Marconi demonstrou como funcionava um tráfego sem fio de um navio para o litoral por meio de código Morse, este por sua vez, patenteou sua invenção em 1896 nos Estados Unidos, ficando conhecido mundialmente como o inventor da

radiodifusão [Held 1999]. O brasileiro patenteou sua descoberta em 1904 nos EUA, mas havia perdido o mérito de inventor.

A tecnologia sem fio é conhecida popularmente por Wi-Fi, termo inglês para *Wireless* Fidelity ou Fidelidade Sem Fio. Embora a conclusão desse termo nunca tenha sido oficializada pela Wi-Fi Alliance, entidade responsável pelo licenciamento dos produtos baseados na tecnologia Wi-Fi [Alliance 2012], também podemos chamá-las de redes 802.11 ou simplesmente redes sem fio. As redes Wi-Fi operam em faixas de frequência que variam de 2.4 a 5.8 GHz e taxa de transmissão que variam de 11 até 70 Mbps com a tecnologia WiMax [Andrews et al. 2007], seus pacotes são transmitidos através do ar, por infravermelho ou em canais de frequência de rádio (radiodifusão) que é mais utilizada [Soares et al. 1995].

As ondas de rádio percorrem caminhos em determinados ambientes para levar o sinal do transmissor ao receptor. Dependendo dos obstáculos encontrados, a potência do sinal pode sofrer repentinas variações, o que pode ser causado pela locomoção de pessoas ou pela posição de obstáculos no ambiente operacional da rede [Kurose e Ross 2006].

Com a evolução da informática, veio também a motivação para melhoria do desempenho das redes *wireless*, que atende necessidades em serviços de celulares, na transmissão de dados via satélite e comunicações sem fio. Ela foi padronizada pelo IEEE como 802.11 *Wireless Local Area Networks Standard Working Group* [Comer 2006] que utiliza DFWMAC como protocolo de controle de acesso ao meio [Soares et al. 1995]. O DFWMAC suporta os métodos de acesso distribuído básico e centralizado, ambos podem coexistir e servem para dar suporte na transmissão do tráfego [Kurose e Ross 2006].

Hoje, muitas empresas, universidades, aeroportos, shopping centers e *cyber-café* utilizam um ponto de acesso (AP, do inglês *Access Point*), local onde se pode estabelecer conexão com a Internet, para dar cobertura aos usuários móveis em uma determinada área de transmissão sem fio. Computadores portáteis como *tablets* e *notebooks*, além de alguns celulares também permitem que essa conexão seja estabelecida.

Uma rede sem fio é composta por um conjunto básico de serviços (BSS – Basic Set Service) que pode conter várias estações base. Uma estação base é responsável pelo envio e recebimento de e para um hospedeiro sem fio que está associado com ela. A estação base se conecta a um dispositivo de interconexão (hub, comutador ou roteador) que “leva” a Internet. Sua arquitetura baseia-se na divisão da área coberta pela rede em células (BSA), o tamanho de cada BSA vai depender das características do ambiente, dos transmissores e receptores [Kurose e Ross 2006].

A Figura 9.1 apresenta uma conexão de sistemas finais (*notebooks*) sem fio entre duas estações que utilizam AP e se comunica com dispositivo de interconexão para transmitir dados pela internet.

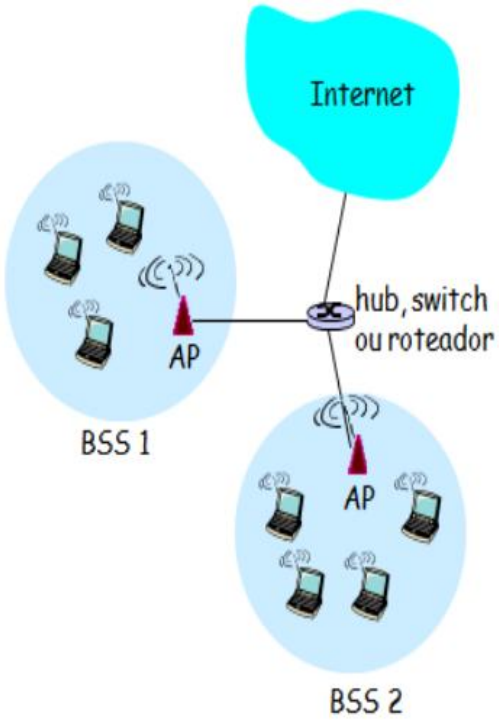


Figura 9.1. Sistema sem fio entre estações [Kurose e Ross 2006**]**

O processo chamado de transferência (*handoff*) acontece quando um hospedeiro móvel se desloca para fora da faixa de alcance de uma estação base e entra na faixa de uma outra, ele muda seu ponto de conexão com a rede maior, ou seja, muda a estação base com a qual está associado.

Através do provedor *wireless*, os clientes que possuem antenas direcionadas, APs e placas de rede sem fio, recebem os serviços disponibilizados pela rede, como compartilhamento de arquivos e serviços de conexão para internet, formando assim uma rede local sem fio. A figura 9.2 ilustra uma estrutura WLAN (*Wireless Local Area Network*) em que empresas, residências e condomínios são clientes que possuem antenas direcionadas para receber sinais através de ondas eletromagnéticas emitidas por um provedor *wireless*.

Redes WLAN podem operar nos modos conhecidos como Infraestruturado e *Ad hoc*. As redes que trabalham no modo Infraestruturado utilizam um AP para interconectar suas estações, quando uma deseja se comunicar com outra, haverá no mínimo dois saltos, onde os quadros são enviados para o AP, que normalmente está conectado a uma rede cabeada tradicional, este por sua vez os envia à estação destino. As redes que trabalham no modo *Ad hoc* também conhecidas como redes independentes permitem que seus clientes se comuniquem uns com os outros em conexões ponto-a-ponto, de modo a compartilhar dados sem a necessidade do AP, para tanto devem estar dentro do raio de alcance da rede [Tanenbaum 2011]. A Figura 9.3 exemplifica uma rede infraestruturada enquanto a Figura 9.4 ilustra a estrutura de uma rede *Ad hoc*.

A autenticação para a comunicação dos clientes seja eles em modo *Ad hoc* ou infraestruturado se deve a um endereçamento. Este endereço possui seis bytes, sendo os

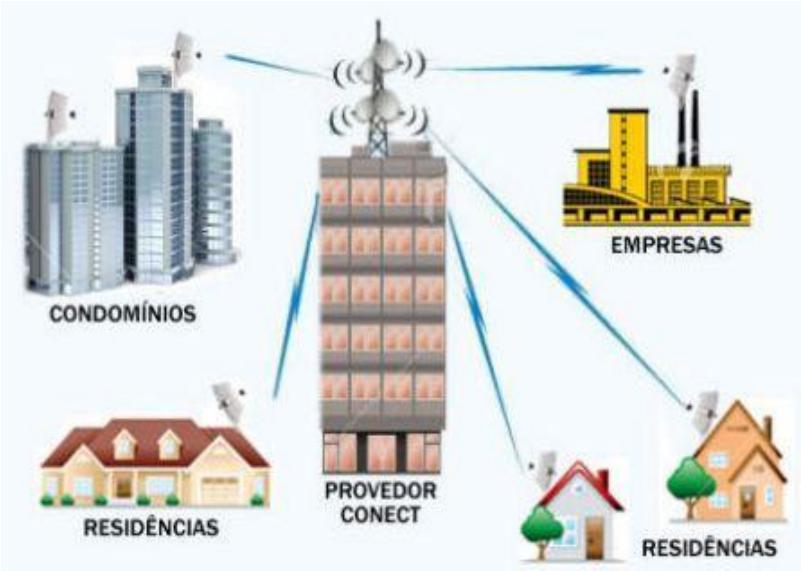


Figura 9.2. Exemplo de uma WLAN

três primeiros para identificação do fabricante e os três bytes seguintes para o número sequencial da placa que serve como numeração única para cada host [Dias and Jr. 2002]. Esta numeração é conhecida como endereço de controle de acesso ao meio - MAC (do inglês *Media Access Control*). Endereços MAC são administrados pelo IEEE e são exclusivos.

A tecnologia de comunicação *wireless* é composta de padrões estabelecidos pelo IEEE, associação que reúne mais de 395.000 associados em mais de 160 países, número esse que cresce constantemente à medida que seus produtos ganham mercado pelo mundo. Os membros constituem-se de engenheiros das áreas de telecomunicações, computação, eletrônica e ciências aeroespaciais, entre outras. O IEEE define normas e padrões tecnológicos ativos e utilizados pela indústria e conta com inúmeros projetos em fase de desenvolvimento [IEEE 2012].

Os padrões utilizados na tecnologia Wi-Fi estão entre os que mais recebem atenção ultimamente e correspondem à família de especificações 802.11, estes padrões especificam as interconexões de computadores, impressoras, dispositivos de vídeo e demais aplicações, através do conceito de rede de comunicação sem fio, ou seja, proporcionam às redes uma comunicação entre um aparelho cliente e uma estação ou ponto de acesso, com o uso de ondas de rádio.

Cada padrão é dotado de características próprias, principalmente no que se refere a modulação, frequência e velocidade de transmissão dos dados. Cada tipo é identificado por uma letra ao final do nome 802.11, alguns exemplos são: 802.11a, 802.11b, 802.11d, 802.11e, 802.11g, 802.11h, 802.11i, 802.11j, 802.11n. Todas estas normas têm o conhecido protocolo Ethernet como protocolo de comunicação entre os dispositivos da rede [Alliance 2012]. A tabela 9.1 apresenta as normas, o nome pelo qual é conhecida e a descrição de cada uma delas.

A figura 9.5 apresenta algumas das características fundamentais dos padrões de enlaces sem fio mais populares. As redes sem fio diferem das redes Ethernet por fatores como: velocidade de

Tabela 9.1. As diferentes normas para tecnologias WiFi

Norma	Nome	Descrição
802.11a	Wifi5	Permite obter um elevado débito (54 Mbps teórico e 30 Mbps real) e especifica 8 canais de rádio na banda de frequência 5 GHz
802.11b	Wifi	Atualmente é a norma mais usada. Propõe um débito teórico de 11 Mbps (6 Mbps real) com um alcance que pode ir até 300 metros em espaço aberto. O intervalo de frequência utilizado é a banda dos 2.4 GHz, com 3 canais de rádio disponíveis
802.11c	Pontage 802.11 versão 802.1d	A norma 802.11c não tem interesse para o grande público. Trata-se unicamente de uma modificação da norma 802.1d a fim de estabelecer uma ponte com as tramas 802.11
802.11d	Internacionalização	A norma 802.11d é um suplemento à norma 802.11 cujo objetivo é permitir uma utilização internacional das redes locais 802.11. Consiste em permitir aos diferentes equipamentos trocar informações nos intervalos de frequência e as potências autorizadas no país de origem do material
802.11e	Melhorias na qualidade do serviço	A norma 802.11e visa dar possibilidades em matéria de qualidade de serviço no nível da camada de enlace de dados. Assim, esta norma tem como objetivo definir as necessidades dos diferentes pacotes em termos de banda concorrida e prazo de transmissão, de maneira a permitir uma melhor transmissão da voz e de vídeo
802.11f	Itinerância (*roaming*)	A norma 802.11f é uma recomendação para vendedores de ponto de acesso para uma melhor interoperabilidade dos produtos. Propõe o protocolo *Inter-Access Point Roaming Protocol* que permite a um utilizador itinerante mudar de ponto de acesso de maneira transparente quando há uma deslocação, independentemente das marcas dos pontos de acesso presentes na infraestrutura rede. Esta possibilidade chama-se itinerância (*roaming*)
802.11g		Oferece um elevado débito (54 Mbps teórico e 30 Mbps real) na banda de frequência dos 2.4 GHz. Tem uma compatibilidade ascendente com a norma 802.11b, o que significa que materiais conformes à norma 802.11g podem funcionar em 802.11b
802.11h		A norma 802.11h visa aproximar a norma 802.11 do padrão Europeu (*HiperLAN 2, doù le h* de 802.11h) e ficar em conformidade com o regulamento europeu em matéria de frequência e economia de energia
802.11i		A norma 802.11h tem como objetivo melhorar a segurança das transmissões (gestão e distribuição das chaves, codificação e autenticação). Esta norma baseia-se no AES (Advanced Encryption Standard) e propõe uma codificação das comunicações para as transmissões que utilizam as tecnologias 802.11a, 802.11b e 802.11g
802.11Ir		A norma 802.11r foi elaborada de maneira a utilizar sinais infravermelhos. Esta norma está agora tecnicamente ultrapassada
802.11j		A norma 802.11j é para o regulamento japonês o que o 802.11h é para o regulamento europeu

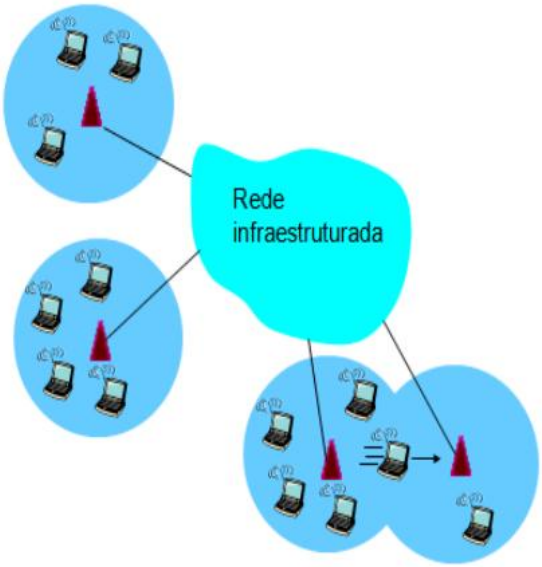


Figura 9.3. Rede infraestruturada

transmissão, área de abrangência, redução na intensidade do sinal, interferência de outras fontes (diferentes fontes podem transmitir na mesma banda de frequência), propagação multivias (porções da onda eletromagnética se refletem em objetos no solo e tomam caminhos de comprimentos diferentes entre um emissor e um receptor) e bits de erros são mais comuns nas redes sem fio. Para amenizar tais problemas é indispensável a utilização de mecanismos de criptografia ao transmitir os sinais, bem como técnicas para espalhamento de banda.

A existência de múltiplos transmissores e receptores sem fio podem criar problemas adicionais como o de terminal oculto e atenuação de sinal. Terminal oculto pode ser caracterizado quando obstruções físicas presentes no ambiente podem impedir a comunicação. A figura 9.6 ilustra um exemplo de terminal oculto em que A consegue se comunicar com B, B se comunica com C, mas não existe conexão entre A e C devido interferência no caminho.

A atenuação de sinal é causada pelo desfalecimento da força de um sinal a medida que ele se propaga no meio sem fio, isso indica que a potência do sinal não é suficiente para chegar a um determinado ponto, mas pode interferir em uma conexão. A figura 9.7 ilustra o caso em que a localização de A e C é tal que as potências de seus sinais não são suficientes para que eles detectem as transmissões de um e de outro, mas, mesmo assim, são suficientemente fortes para interferir uma com a outra na estação B.

9.3. Protocolos de comunicação

Existe uma pilha de protocolos encarregada para o envio e recebimento de dados na rede. Esse conjunto de protocolos é representado por dois protocolos bem conhecidos, o Protocolo de Controle de Transmissão – TCP (do inglês *Transmission Control Protocol*) e o Protocolo da

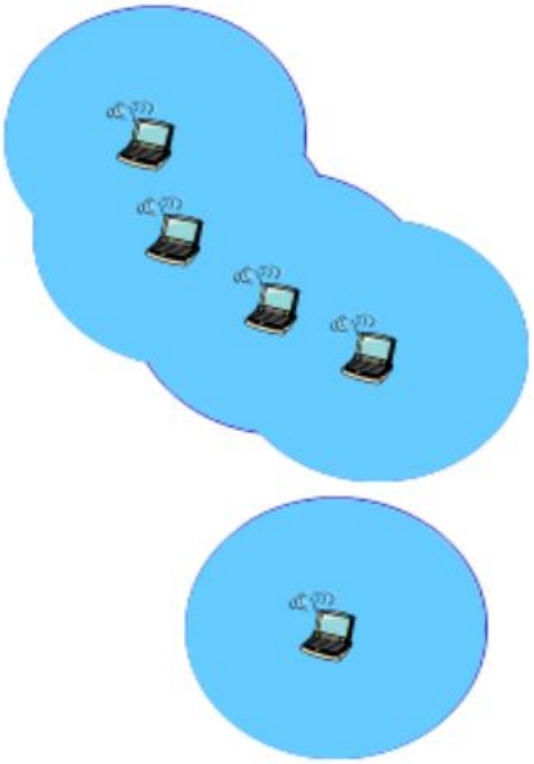


Figura 9.4. Rede *Ad hoc*

Internet (IP - *Internet Protocol*) que juntos formam o TCP/IP. Este por sua vez é agrupado em quatro camadas, como ilustrado na figura 9.8.

A camada de Aplicação é responsável pela comunicação entre os aplicativos da Internet para o protocolo de transporte, ou seja, ela define os protocolos utilizados nas aplicações dos usuários. Os principais protocolos dessas aplicações são: FTP, TELNET, SMTP, DNS e HTTP.

A camada de Transporte transforma as informações obtidas da camada de aplicação em pacotes, os quais serão roteados e endereçados ao seu destinatário através da camada de Internet. Nesta camada são utilizados os protocolos TCP e UDP Protocolo de Datagrama de Usuário (*User Datagram Protocol*).

A camada Internet recebe os pacotes da camada de transporte, faz seu endereçamento e controle do envio e destino dos pacotes e repassa estes pacotes para a camada de rede. Sua comunicação entre as camadas é estabelecida através de datagramas.

A camada de Host ou Rede faz a abstração de hardware, isto é, faz a comunicação dos pacotes IPs entre os vários tipos de tecnologias de redes existentes.

Os pacotes IPs serão completamente preenchidos ao final da passagem por todas as camadas, a presentado uma sequência lógica de bits. A Figura a seguir ilustra todos os campos de um pacote IPv4 bem como o número de bits disponíveis para cada campo.

O campo Versão indica a versão do protocolo IP, atualmente utiliza-se a versão 4, ou seja, IPv4, para verificar a validade do datagrama. O campo comprimento de cabeçalho indica o número de palavras de 32 bits que constituem o cabeçalho do datagrama, onde

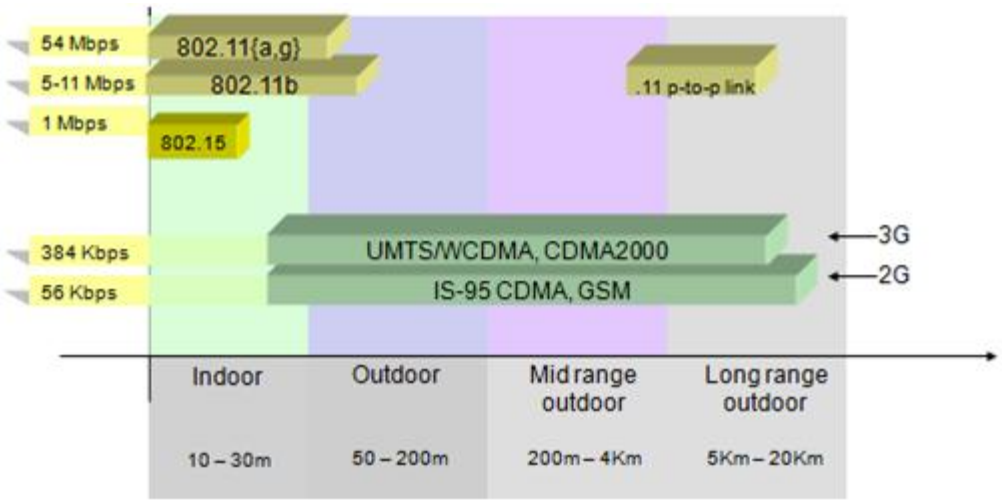


Figura 9.5. Características de enlaces de padrões selecionados de rede sem fio [Kurose e Ross 2006]

Figura 9.6. Terminal oculto [Kurose e Ross 2006]

o valor mínimo é 5, ou seja, um cabeçalho de um datagrama é representado pelos quatro primeiro campos. A soma desses campos corresponde a uma palavra de no máximo 32 bits formando o cabeçalho do datagrama, assim o tamanho desse cabeçalho é representado por 4 bits pelo campo comprimento do cabeçalho. A maneira como o datagrama deve ser tratado deve ser indicado no campo Tipo de serviço. A dimensão total do datagrama em bytes, com um mínimo de 2 bytes e um máximo de 65.536 bytes é indicada no campo Comprimento total.

O campo Duração de vida indica o número máximo de switchs através dos quais o datagrama pode passar. Este campo é reduzido a cada passagem do pacote por um switch. Quando este atinge o valor de 0, o switch destrói o datagrama. Com isto, evita-se o congestionamento da rede pelos datagramas perdidos. O protocolo que o datagrama utiliza

é identificado no campo Protocolo, podemos ter: ICMP, IGMP, TCP ou UDP. Para controlar a integridade do cabeçalho a fim de determinar se este não foi alterado durante a transmissão o campo Checksum do cabeçalho contém um valor codificado de 16 bits.

O campo Identificação permite a fragmentação dos datagramas, é identificado por ordem e números. O campo Flags contém seis flags (sinalizadores). Os flags SYN e FIN são usados, respectivamente, no estabelecimento e no término de uma conexão. ACK é ativado quando o campo número de confirmação tiver que ser considerado. URG é ativado quando o segmento contém dados urgentes (o campo ponteiro de urgência indica onde começam os dados não urgentes contidos no segmento). A ativação de PSH indica que o receptor deve entregar os dados à aplicação mediante sua chegada, em vez de armazená-

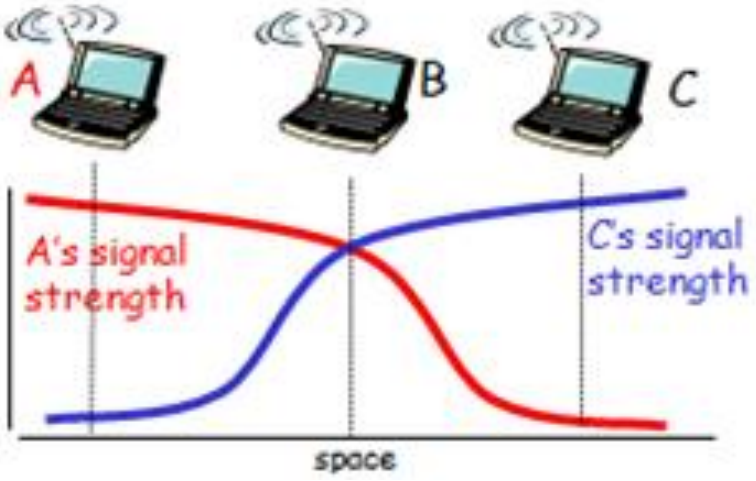


Figura 9.7. Atenuação do sinal [Kurose e Ross 2006]

Figura 9.8. Modelo de camadas do protocolo TCP/IP

los até que um buffer completo tenha sido recebido. RST é utilizado para reiniciar uma conexão que tenha ficado confusa por algum motivo. Os campos Endereço IP fonte e Endereço IP destino representam o endereço IP da máquina emissora, para que seja possível ao destinatário responder e o endereço IP do destinatário da mensagem, respectivamente. As opções são destinadas a alterações (caso haja). Por fim, o campo Dados é destinado a informações dos usuários das aplicações.

Na camada de Rede é estabelecida a comunicação entre os hosts. Para essa comunicação a camada é dividida em três outras subcamadas, são elas: Controle de Link Lógico LLC *(Logical Link Control*), Controle de Acesso ao Meio MAC (*Media Access Control*) e Física PHY (*Physical*) como ilustrado na figura 9.10, observem como são expostas as subcamadas e sua relação entre os diferentes tipos padrões.

A subcamada LLC é padronizada pelo IEEE 802.2. Sua função é colocar todas as informações na forma de cabeçalho estabelecendo uma ordem de sequência e confirmação. As subcamadas MAC e PHY têm como função estabelecer a divisão entre os padrões, fazendo a comunicação entre os meios físicos distintos, como exemplos o padrão cabeado estabelecido pelo IEEE 802.3 e o padrão sem fio, estabelecido pelo IEEE 802.11.

O Enlace dos quadros é criado através da ponte entre LLC e o MAC, constituindo assim os frames com cabeçalho, que serão enviados por algum padrão do meio físico. O

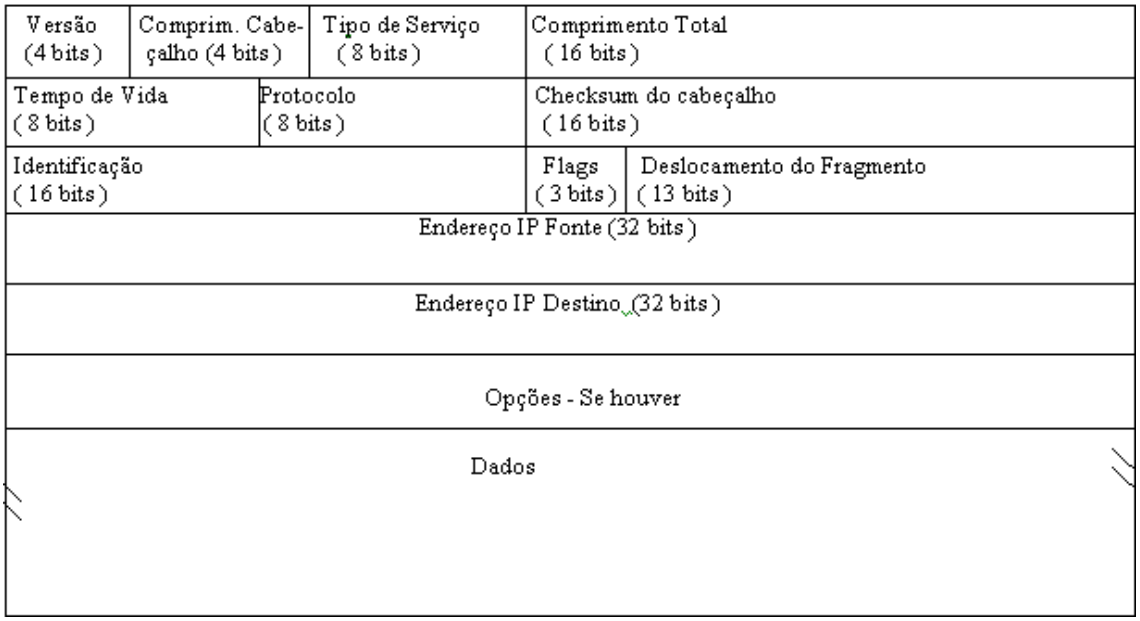


Figura 9.9. Corpo do pacote IPv4

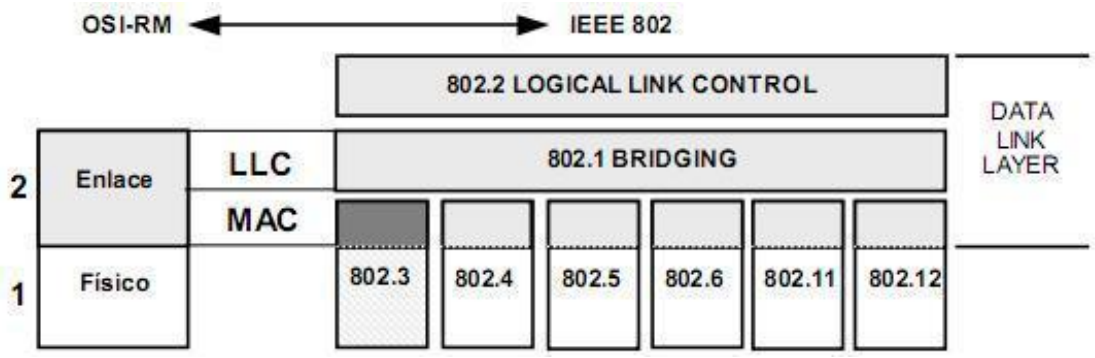


Figura 9.10. Relação das subcamadas e os padrões 802 [Calazans et al. 2001]

meio físico se constitui dos diferentes padrões:

- 802.3 - Fazem o controle de acesso ao meio com detecção de colisão de dados, baseado no CSMA-CD (Carrier Sense Multiple Access with collission Detection).

- 802.4 e 802.5 - Especifica o tipo de protocolo de permissão de passagem, que pode ser implementado em barramento (token bus) ou em anel (token ring).
- 802.6 - Define o padrão para o transporte de dados de alta velocidade, como utilizados em redes metropolitanas.
- 802.11 - Especifica o padrão para o ambiente sem fio, criando conexões de redes entre os equipamentos sem fio, em uma área local.
- 802.12 - Especifica a comunicação baseada em uma topologia de rede em estrela, onde se faz necessário a utilização de um concentrador, um dispositivo denominado hub, que tem como função conceder permissão de transmitir entre as estações.

Com base na comunicação da última camada do TCP/IP (camada de rede) entre os hosts, todo o enlace dos dados é criado entre a comunicação de suas subcamadas, assim seus campos que constituem seu cabeçalho são preenchidos e determinados com uma palavra de no máximo 32 bits. Esse cabeçalho, também chamado de frame, será enviado para o host de destino juntamente com o restante dos bits que formam o corpo de um quadro IP (como apresentado na figura 1.9) os quais foram preenchidos pelas outras camadas do TCP/IP. O quadro ou pacote IP (conjunto de frames), contém todos os bits necessários para conexão com o host destinatário, e as informações produzidas pelo usuário do host emissor, que ao receber terá todo procedimento de leitura dessa sequência de bits.

A leitura do host receptor fará a validação desses bits, ou seja, comparar se os bits contêm seu endereço, se não contêm erros de transmissão, o tipo de protocolo usado e assim por diante, até a validação de todos os campos necessários que corresponde ao quadro IP.

Quando validado os campos, é então conhecida a integridade das informações e assim repassadas às camadas subsequentes do TCP/IP como um processo inverso ao do host emissor. Essas camadas farão a recomposição dos quadros IPs, abstraindo os dados do campo de dados e formando assim a informação completa, a qual será entregue ao usuário receptor pela última camada (camada de aplicação).

9.4. Segurança

Quando falamos em segurança de redes é importante ressaltar que ela é conseguida por meio de um ciclo contínuo de proteção, detecção e reação, além disso, é desejável que algumas propriedades como confidencialidade, autenticação, integridade e disponibilidade sejam implementadas para se ter uma comunicação segura.

A confidencialidade garante que somente o remetente e o destinatário devem poder entender o conteúdo da mensagem transmitida, isso pode ser conseguido através de técnicas de cifragem e decifragem. A autenticação serve para confirmar que as partes envolvidas realmente são quem alega ser. A integridade permite assegurar que o conteúdo da mensagem não foi alterado durante a transmissão, seja por acidente ou por má intenção. Disponibilidade garante que as informações estarão disponíveis aos usuários quando estes precisarem.

Um dos grandes problemas das redes sem fio está relacionado à segurança da radiodifusão. É possível que receptores não autorizados possam captar o sinal em sua área de abrangência e assim utilizar de forma indevida as informações captadas na rede.

A segurança da informação é a principal questão nas redes Wi-Fi. Como o sinal é propagado no ar, todos os pacotes, inclusive os que contém informações como números de cartão e senhas bancárias de clientes, trafegam pelo ar de forma que podem ficar expostos. Assim, invasores que desejam obter tais informações podem utilizar determinadas técnicas e softwares para tentar capturar os pacotes da rede.

Ao tempo que as redes sem fio oferecem como vantagens a mobilidade de seus usuários, devido a ausência de fios e a possibilidade das informações chegarem a locais de difícil acesso, sua estrutura deixa a desejar no quesito segurança, por não haver proteção em relação ao meio por onde as informações trafegam, no qual não há nenhuma proteção física.

À medida que se tem qualquer meio de comunicação entre as estações, a rede se torna vulnerável à perda de pacotes devido à distorção ou interferência do sinal, ou ainda pior, a ataques de pessoas mal intencionadas. Falta de conhecimento, segurança e planejamento, podem trazer grandes riscos para empresas e pessoas que preferem utilizar essa tecnologia. Uma das maiores vulnerabilidades existentes em redes sem fio são criadas pela má configuração do equipamento.

Técnicas de filtragem configuradas no AP ou no roteador para garantia de confiabilidade na conexão e técnicas de encriptação de dados que garantem a integridade das informações foram padronizadas pelo IEEE 802.11 com o objetivo de melhorar a segurança nas redes WI-FI. Como exemplos de técnicas de filtragem temos identificação pelo SSID e Filtragem por endereço MAC. E as de encriptação são WEP, TKIP, WPA e WPA2. Todas serão descritas a seguir.

9.4.1. SSID Service Set Identifier

O SSID é um nome de identificação da rede, podendo ser criado pelo administrador ou mantido o nome padrão que é estabelecido automaticamente durante a configuração do equipamento [de Morais Jardim 2007].

Para se conectar a uma rede 802.11 é necessário saber seu SSID, que corresponde a um dos quesitos para efetivar a autenticação dos clientes. No entanto, o SSID pode ser descoberto no momento em que se verifica a existência de redes sem fio. Por padrão, o SSID é parte do cabeçalho enviado por todos os pacotes através da WLAN.

Uma medida de segurança que pode ser utilizada para evitar a identificação da rede por intrusos, seria desabilitar o modo broadcast do SSID. Tal estratégia evitaria que o AP anuncie a rede, aumentando assim o nível de dificuldade para invasores, que teriam problemas em conhecer o identificador da rede a qual se deseja associar.

9.4.2. Filtragem por endereço MAC

Cada adaptador de rede, também chamado de cartão de rede, possui um endereço físico atribuído pelo seu fabricante. Este endereço é composto de 12 números hexadecimais, agrupados e separados por pares e é conhecido popularmente como endereço MAC [Peterson and Davie 2004].

Os pontos de acesso ou roteadores permitem uma configuração onde podem gerir uma lista de direitos de acesso. Essa configuração é definida com o cadastro do endereço

MAC de cada cartão de rede dos usuários, permitindo limitar o acesso de várias máquinas na rede e restringindo o acesso apenas para os usuários cadastrados.

9.4.3. WEP Wired Equivalent Privacy

O protocolo WEP, acrônimo de Wired Equivalent Privacy que significa privacidade Equivalente Sem fio, fornece autenticação e criptografia de dados entre um hospedeiro e um ponto de acesso sem fio utilizando uma abordagem de chaves simétricas compartilhadas. Este protocolo é encarregado da codificação dos frames através do algoritmo simétrico RC4 que faz a criptografia utilizando chaves de comprimento de 64 bits ou 128 bits. Ele está embutido no padrão 802.11b [Whalen 2002].

O WEP não especifica um algoritmo de gerenciamento de chaves, portanto, se admite que o hospedeiro e o ponto de acesso sem fio concordaram com uma chave por meio de um método fora da banda. A autenticação é realizada da seguinte forma: primeiro um hospedeiro sem fio requisita autenticação por um ponto de acesso, em seguida o ponto de acesso responde à requisição de autenticação com um valor de *nonce* de 128 bytes, depois o hospedeiro sem fio criptografa o *nonce* usando a chave simétrica que compartilha com o ponto de acesso e por fim, o ponto de acesso decripta o *nonce* criptografado pelo hospedeiro. Se o *nonce* decriptado corresponder ao valor enviado originalmente pelo hospedeiro, então este será autenticado pelo ponto de acesso [Kurose e Ross 2006].

O WEP se restringe a uma chave secreta de 64 ou 128 bits, onde, por padrão, 24 bits correspondem a uma chave fixa com o nome de IV (vetor de inicialização). A chave secreta é declarada e configurada no ponto de acesso e nos clientes. O algoritmo utilizado de criptografia, no caso o RC4, cria através da chave um número pseudo-aleatório com comprimento igual ao do frame. Assim, cada transmissão de dados é calculada utilizando um número pseudo-aleatório como mascaramento.

O algoritmo criptográfico WEP é ilustrado na figura 9.11. Admite-se que ambos, um hospedeiro e o ponto de acesso, conhecem uma chave simétrica secreta de 40 bits, Ks. Além disso, um vetor de inicialização (IV) de 24 bits é anexado a chave de 40 bits, criando uma chave de 64 bits que será utilizada para criptografar um único quadro. O IV mudará de um quadro para outro e, por conseguinte, cada quadro será criptografado com uma chave de 64 bits diferente. A criptografia é efetuada como segue. Em primeiro lugar, é calculado um valor de CRC de 4 bytes para a carga útil. A carga e os 4 bytes do CRC então são criptografados usando o codificador sequencial RC4. Quando recebe um valor de chave (nesse caso, a chave de 64 bits (Ks, IV)), o algoritmo RC4 produz uma sequencia de valores de chaves k1IV, k2IV, k3IV, ...que são utilizados para criptografar os dados e o valor de CRC em um quadro [Kurose e Ross 2006].

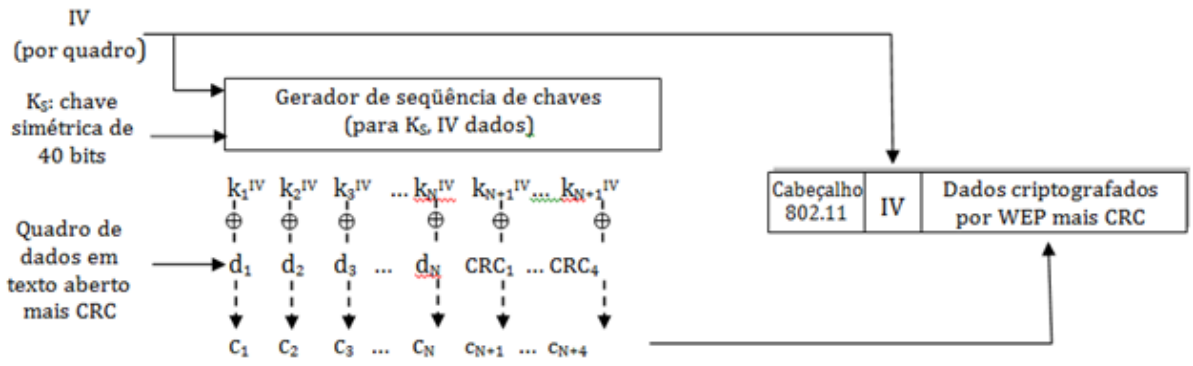


Figura 9.11. Protocolo WEP 802.11

9.4.4. TKIP Temporal Key Integrity Protocol

O Protocolo de Integridade de Chave Temporal TKIP, foi criado com a finalidade de contornar os problemas do WEP. Também utiliza o protocolo RC4 para cifrar os dados. Além disso, utiliza uma chave por pacote (*per-packet key mixing*), onde cada estação combina a sua chave com seu endereço MAC e assim é criada uma chave criptografada que é única. Essa chave é compartilhada entre o ponto de acesso e os clientes *wireless*.

O TKIP permite a geração aleatória de chaves e oferece a possibilidade de alterar a chave de forma dinâmica automaticamente [Prodanovic and Simic 2007]. Assim, fica mais difícil para um atacante quebrar a chave e mesmo que ele consiga, a vida útil da chave será pequena.

9.4.5. WPA e WPA2 Wi-Fi Protected Access

Pode-se dizer que o WPA é uma otimização do WEP, que possui protocolos de autenticação e um algoritmo de cifragem robusto como o TKIP . A grande vantagem do WPA é a melhoria no processo de autenticação de usuários. Essa autenticação se utiliza do protocolo 802.11x e do EAP (Extensible Authentication Protocol) que através de um servidor de autenticação central faz a autenticação de cada usuário antes deste ter acesso à rede.

O WPA conta com tecnologia aprimorada de criptografia e de autenticação de usuário. Cada usuário tem uma senha exclusiva que deve ser digitada no momento da ativação do WPA. No decorrer da sessão a chave de criptografia será trocada periodicamente e de forma automática, graças aos serviços do TKIP, que também fazem parte do protocolo WPA. Assim, torna-se significativamente mais difícil que um usuário não-autorizado consiga se conectar à rede. A desvantagem do WPA em sua primeira versão é o suporte apenas às redes em modo infraestruturado, o que significa que não permite proteção em modo *Ad hoc*.

O WPA2 é a evolução WPA e com melhorias adicionadas ao padrão 802.11i, como por exemplo, a utilização de um algoritmo forte de criptografia, o AES (Advanced Encryption Standard). O AES permite a utilização de chaves de 128, 192 e 256 bits, constituindo assim uma ferramenta poderosa de criptografia. Além disso, seu uso pode ser aplicado tanto em redes de infraestrutura como a de modo *Ad hoc*

9.5. Invasão

A praticidade da implantação de serviços *wireless* faz com que, em alguns casos, administradores de redes não atendam os meios necessários para uma segurança básica. Em muitos casos a rede é configurada utilizando o padrão definido pelo fabricante, facilitando o ataque de invasores.

A existência dessa fragilidade faz com que um intruso possa ter acesso não autorizado ao computador ou à informações que se propagam no ar, podendo ele comprometer a disponibilidade do sistema, a integridade ou a confiabilidade dos dados. Dessa forma, o intruso pode ter comportamentos diferentes em relação às informações. Como exemplo: Interromper o fluxo entre origem e destino das mesmas, fazer a leitura para se ter conhecimento das informações, modificar seu conteúdo ou até mesmo interferir totalmente na origem e destino das informações.

A rede sem fio em muitos casos é controlada por um AP. Esse AP faz a conexão de ponte entre os dispositivos sem fio e a rede cabeada, possibilitando que os dispositivos sem fio tenham os mesmos serviços que a rede cabeada. Como exemplo, tem-se a conexão com a internet. Contudo, à medida que a rede sem fio tem seu sinal propagado no ar, tanto os dispositivos habilitados quantos os que não são, fazem leituras desse sinal.

Com isso, o intruso por estar dentro do raio de frequência, também pode fazer uso das informações e serviços da rede, obtendo acesso a essa rede ou utilizando-se das técnicas existentes para capturar as informações que estão sendo trafegadas no ar. Duas dessas técnicas são bem populares, como é o caso dos *Wardriving* e *Warchalking*, a seguir descritas:

9.5.1. *Wardriving*

O nome *Wardriving* foi escolhido por Peter Shipley, para definir atividade de dirigir um automóvel à procura de redes *wireless* abertas, passíveis de invasão [Castro Peixoto]. Para efetuar a prática do *Wardriving* são necessários um automóvel, um computador, e uma placa Ethernet configurada no computador. No computador é feita a captura e a leitura dos pacotes, a partir dos quais é extraído o conhecimento a respeito das redes existentes naquele local. Essa atividade não é maliciosa desde que não haja violação das informações, pois, alguns se contentam apenas em encontrar as redes desprotegidas, enquanto outros se obtêm do uso, o que já ultrapassa os níveis permitidos da atividade.

9.5.2. *Warchalking*

O nome *Warchalking* foi inspirado na época da grande depressão norte-americana, quando andarilhos desempregados criaram uma linguagem de marcas de giz ou carvão em vários lugares da cidade, indicando assim, uns aos outros o que esperar de tal lugar, como comida e abrigo temporário. Então, *Warchalking* é a prática de escrever símbolos indicando a existência de redes *wireless* e informando sobre suas configurações [Castro Peixoto]. As marcas usualmente feitas em giz em calçadas indicam a posição de redes *wireless*, facilitando a localização para uso de conexões alheias pelos simpatizantes da ideia. Tais símbolos são descritos na figura 9.12.

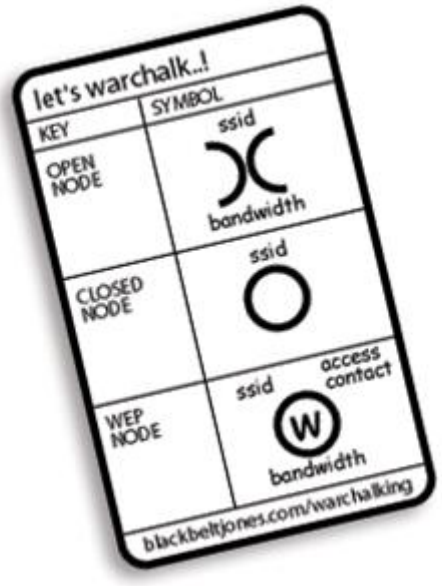


Figura 9.12. Símbolos utilizados pelos *Warchalking*

Os significados de cada símbolo da figura são:

- *Open Node* significa que a rede está aberta e vulnerável;
- *Closed Node* indica que a rede é uma rede fechada;
- *WEP Node* que possui a letra W dentro do círculo e quer dizer que a rede *wireless* utiliza o padrão de segurança WEP *Wireless Equivalent Privacy*, ou seja, com presença de criptografia.

Em cima de cada símbolo, tem-se o SSID (*Service Set Identifier*) que é o nome de identificação da rede para login. Essas informações podem ser obtidas através de software próprio conhecidos como *Sniffers* [Castro Peixoto].

9.5.3. *Sniffers*

Sniffers são programas capazes de capturar pacotes de informações que se propagam na rede. Seu propósito legal é analisar o tráfego da rede e identificar áreas potenciais de preocupação [Dhar]. Mas, seu uso pode ser aplicado de maneira ilegal, fazendo com que sejam feitas varreduras na rede para capturar o tráfego e assim obter senhas ou informações confidenciais, ou até mesmo para abrir lacunas na segurança de redes vizinhas com o objetivo de obter acesso não autorizado.

9.5.4. Tipos de ataques

Pode-se dizer que existem vários tipos de intrusos que usam diferentes técnicas de ataques a fim de capturar dados nas redes sem fio. Esses ataques podem acontecer desde a forma mais simples até a mais complexa. Os ataques se diferenciam da seguinte forma:

- Ataque de inserção: são aqueles em que o atacante tenta associar dispositivos não autorizados na rede iludindo os meios de segurança, informando que esses dispositivos já fizeram a autenticação.
- Ataque de monitoração: um atacante pode monitorar o tráfego da rede e assim capturar as informações e analisá-las. Para isso, faz-se uso de ferramentas *Sniffers*.
- Ataque de associação maliciosa: um atacante pode se passar por um AP, que simula como se fosse um ponto de acesso de uma rede sem fio real, fazendo com que o cliente se conecte a ele e assim consegue obter as informações do cliente.
- Ataque de ARP *poisoning*: quando o atacante utiliza uma técnica de envenenamento do protocolo de endereços, que funciona na camada de enlace e faz com que limite os serviços da vítima. Este ataque só pode ser feito se o atacante estiver conectado na mesma rede que a vítima.
- MAC spooffing: algumas redes podem estar configuradas limitando os usuários por cadastro do endereço MAC dos seus dispositivos. Dessa maneira, essa técnica consiste em obter um endereço válido de um cliente cadastrado e substituir seu endereço MAC pelo endereço do cliente e assim se passar por um cliente válido na rede.

- Ataque de D.O.S.: esse é o ataque de negação de serviço (*Denail of Service*) que é capaz de deixar algum serviço ou recurso da rede indisponível. Podem ser feitos tanto pela utilização da rádio frequência de eletrodomésticos, por funcionarem na

mesma frequência da rede sem fio, causando ruídos ou distorção do sinal, como também pelas técnicas já citadas. Como exemplo, ataque de associação maliciosa e MAC *spooffing*, que permite se passar por algum cliente válido ou um ponto de acesso existente.

- Ataques de vigilância: para muitos não se trata de nenhum problema, pois consiste em percorrer lugares em busca de observar sinal existente de redes sem fio e assim conhecer esses pontos; os ataques de vigilância são exemplos dos já citados *Wardriving* e *Warchalking*.

9.6. Técnicas que podem Burlar a Segurança

Após o lançamento do padrão 802.11, também foi incluído um mecanismo para a segurança das informações. Esse mecanismo garante que as informações não sejam violadas para isto existe o uso de criptografia de dados. Em sua primeira versão, ele foi denominado de WEP. Mas, com o crescimento exponencial das WLANS, foram necessários melhores mecanismos de criptografia. Essa necessidade se deu ao serem detectadas as fragilidades no WEP. Assim, surgiram as novas versões como WPA e WPA2.

A missão do WPA e WPA2 é contornar os problemas da primeira versão, com o uso de novas técnicas para garantir a integridade das informações. Como exemplo, o EAP, para o qual se faz necessária a autenticação entre o cliente e o ponto de acesso, o TKIP, que possibilita que sua chave de criptografia possa ser mudada dinamicamente e por fim, o AES, que é um algoritmo bem mais forte que o RC4, utilizado no WEP e WPA.

Independente de haver mecanismos de segurança ou não, os usuários de rede sem fio utilizam os seus serviços para ter uma estrutura mais prática que permite facilidade de uso e mobilidade . Em geral, os usuários estão mais preocupados em ter acesso e utilizar a rede e na maioria das vezes não se preocupam com a implementação dos mecanismos propostos para aumentar a segurança. Assim, muitas vezes são utilizados mecanismos não tão eficazes que deixam dados pessoais vulneráveis a ataques.

De acordo com os dados obtidos, serão estudadas as fragilidades existentes nos meios utilizados pelos administradores como forma de segurança de suas WLANs. A forma a ser abordada será por meios de ferramentas ou meios necessários que relatam ou que comprovam não só na teoria, como é visto na maioria dos trabalhos publicados, mas também na prática, mostrando a fragilidade na segurança de cada um dos mecanismos.

O estudo de técnicas para burlar a segurança, será apenas para o entendimento e análise geral de como usuários mal intencionados fazem na prática os seus ataques. Em síntese, a idéia é, a partir dessa análise, fazer com que novos estudos sejam feitos para que se obtenham meios de defesa contra supostos intrusos.

A ordem a ser estudada de como pode se burlar os mecanismos de segurança será da seguinte forma: como burlar o SSID, como burlar o MAC, como burlar o WEP, como burlar o WPA e como burlar o WPA2.

9.6.1. Burlando o SSID

O SSID pode ser facilmente encontrado utilizando ferramentas adequadas. O NetStumbler é uma ferramenta de Scanners que consegue encontrar o SSID oculto. Ou seja, mesmo que os administradores de WLANS desativem o modo de Broadcast, fazendo que o ponto de acesso não envie informações do SSID, o NetStumbler o encontra. Isso ocorre devido o

NetStumbler ser capaz de enviar pacotes para todas as direções no objetivo de associar o nome da rede ao qual está oculta.

Com objetivo de analisar esta técnica, será provado na prática esse comportamento com o NetStumbler. Para isso, será criada uma rede infraestrutura, e serão utilizados dois *notebooks* (por terem placa de rede sem fio embutida) e um computador de mesa conectado à internet por cabo. Os três computadores estão configurados com sistema operacional Windows XP. Já no computador de mesa foi instalada uma placa de rede sem fio Ralink. E compartilhando à internet para essa placa. O software Ralink permite em sua configuração ser colocada em modo de ponto de acesso. Neste caso, a placa utilizada fará o mesmo trabalho de um roteador (AP) e assim servirá de ponto de acesso para os clientes (os *notebooks*). A figura 9.13 detalhada a configuração do software da placa Ralink.

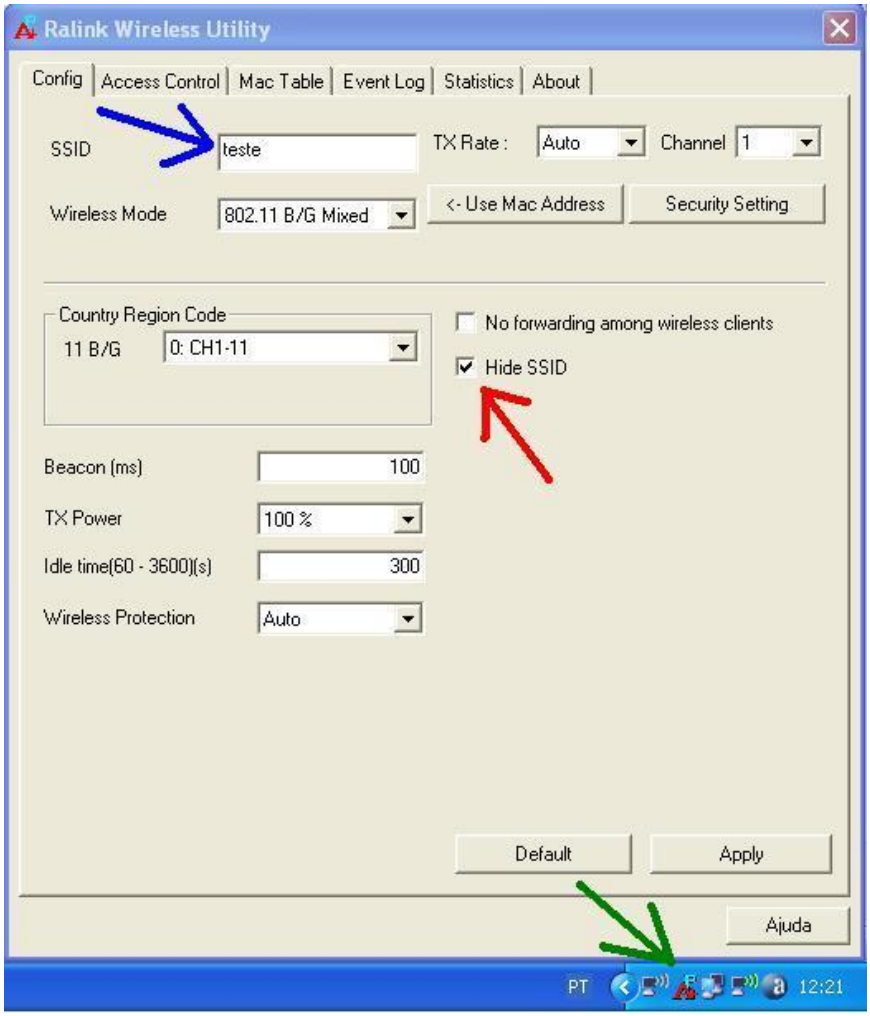


Figura 9.13. Software da Placa Ralink em modo de AP

Para criar o ponto de acesso só serão feitos dois passos simples no menu config da interface. O campo de texto para SSID foi nomeado de teste. Este indica então o nome de identificação da rede e está indicado com a seta azul. A seta vermelha indica que foi marcada a opção de ocultar SSID. Já a seta verde indica na barra de tarefas do Windows que o software foi posicionado em modo AP e assim tem-se um ponto de acesso esperando por clientes.

Em seguida, em um dos *notebooks* foi instalado e executado o NetSumbler. A figura 1.14 ilustra o funcionamento da ferramenta. Sua interface está exibindo duas colunas que serão suficientes. Uma com o nome de MAC, onde indica o endereço MAC dos pontos de acesso e outra com o nome de SSID, onde indica o nome do ponto de acesso. Então, posto o NetStumbler em execução foram detectadas três redes.

Duas dessas redes são redes vizinhas com o nome do SSID riscado, por não haver necessidades de serem exibidas. A terceira é uma rede onde o SSID está em branco, significando que o SSID está oculto com o MAC correspondente a 000E2E4A2B6D. Então, esta é a rede criada com a placa instalada no computador de mesa e com o software Ralink como ilustrado na figura 9.14. Como pode ser visto, o NetStumbler não detectou o SSID. Mas, isso se deve ao fato de não haver nenhum cliente na rede se comunicando com o ponto de acesso. É neste ponto que entra a necessidade do segundo notebook, que servirá de cliente na rede.

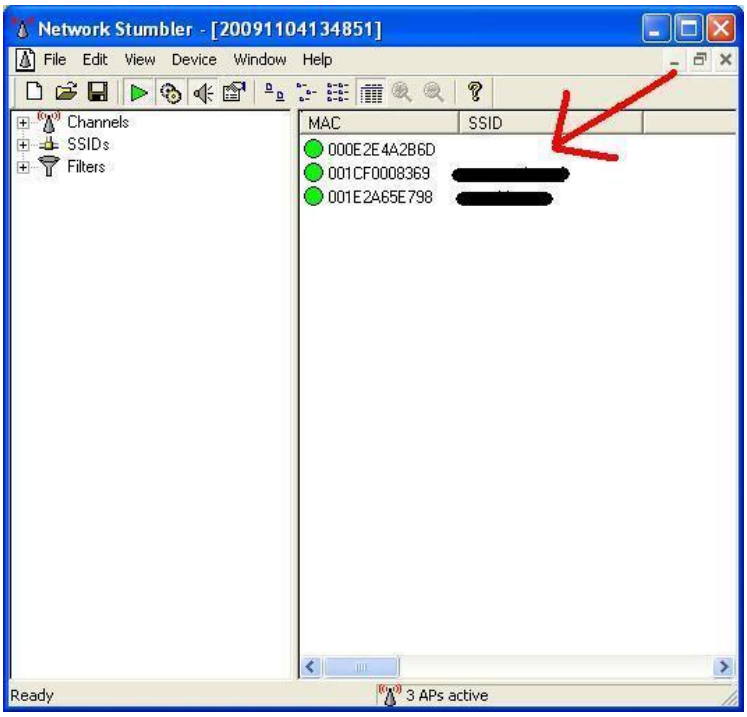


Figura 9.14. NetStumbler em execução

Dessa maneira, para se conectar na rede como cliente será necessário o conhecimento do SSID da rede. Como já é conhecido previamente que o nome da rede é “teste”, então o segundo notebook foi configurado para se conectar à rede que corresponde ao SSID "teste". Isso foi feito por que a rede está oculta, então o Windows XP não detecta tal rede. Portanto, tem que se informar manualmente o SSID, para que o sistema operacional detecte a máquina na rede. Mesmo as versões mais modernas do sistema operacional Windows ou mesmo outros sistemas operacionais não detectarão a rede. O máximo que eles detectarão será uma rede e mais algumas informações, como exemplo, o nível de sinal. Mas, eles não detectarão o seu respectivo SSID pela simples razão de ele estar oculto. Então, feito o segundo notebook se conectar à rede, percebeu-se, no

NetStumbler em execução, que ele detectou o SSID, como mostrado e indicado pela seta vermelha da figura 9.15.

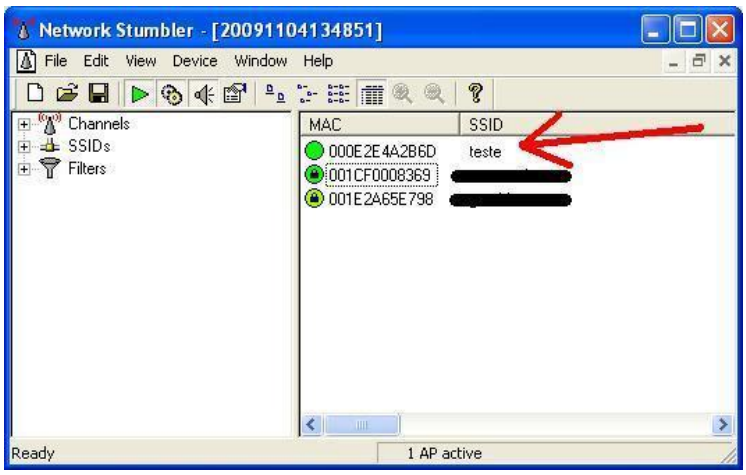


Figura 9.15. NetStumbler detectando o SSID oculto

O campo que estava em branco, antes oculto, agora exibe o nome da rede. Isso se deu no momento em que o cliente se autenticou com o ponto de acesso. Dessa maneira, durante as trocas de informações para autenticar um cliente, o NetStumbler em execução, por enviar pacotes em todas as direções, também detecta os clientes que estão a se conectar. Como o cliente enviou o nome do SSID para fazer a conexão, o NetStumbler comparou com qual MAC de ponto de acesso ele se autenticou, tendo então, através do cliente, a revelação do SSID.

9.6.2. Burlando o MAC

Burlar o MAC seria o mesmo que informar outro endereço físico para placa de rede sem fio. No momento em que isso for feito, utilizando um endereço já existente de outra placa, dá-se então a ação conhecida como clonagem do MAC. Para praticar esse ato, é necessário que o driver da placa, que está instalada, permita este recurso. Como exemplo, os drives das placas da fabricante Ralink para o sistema operacional Windows XP.

O procedimento, para clonar o MAC, pode ser feito no próprio sistema operacional Windows XP ou com a utilização de algumas ferramentas. Nos exemplos, será mostrado como ocorre a clonagem nos dois modos. A clonagem pelo Windows XP pode ser feita abrindo o gerenciador de dispositivos com os seguintes passos: menu Iniciar > Painel de Controle > Sistema. Depois no menu Hardware e sub-menu Gerenciador de dispositivos. Após isso, será aberta uma janela como ilustrado na figura

Na janela ilustrada na figura 9.16, são exibidos todos os dispositivos existentes no computador. Indo em adaptadores de rede, visualizam-se os dispositivos de redes que estão instalados, como indicado na seta vermelha. Como pode ser visto, existe um cartão de rede sem fio instalado. Clicando nesse dispositivo de rede sem fio, obtém-se acesso às propriedades do dispositivo. Então no menu Avançado da janela de propriedades será exibia uma lista de nomes de propriedades deste dispositivo, como ilustrado na figura

Figura 9.16. Gerenciador de dispositivos

1.17. Na lista de propriedades, há um campo com o nome: Local Administation MAC

Network, indicado pela seta verde. Porém, esse campo pode ser descrito com outro nome, dependendo do driver e cartão que esteja instalado. Entretanto, quase sempre é seguido com o nome: MAC ou MAC Address. Então, após escolher esse campo, ter-se-á o acesso ao campo Valor, como indica a seta vermelha. Portanto, é nesse campo que podem ser inseridos os 12 números hexadecimais que correspondem ao endereço MAC. E, dessa maneira, substituir o endereço padrão da placa.

Outro método mais simples e executado também no Windows XP é a utilização de ferramentas, que no exemplo será mostrado com a ferramenta MacMakeUp. Essa ferramenta é um executável, após cuja execução uma janela será aberta onde três passos simples devem ser realizados. Na figura 1.18 é exibido o MacMakeUp em execução.

Os três passos correspondem às três setas na figura, onde a seta vermelha indica o lugar onde se escolhe qual placa deseja para fazer a alteração. Após escolher qual a placa, no campo New address, indicado com a seta verde, será onde deverão ser escrito os 12 caracteres hexadecimais que irão substituir o número padrão da placa. Após isso, só faltará a conclusão clicando no botão Change indicado com a seta azul. Depois desses procedimentos, para que o novo MAC Address seja válido, basta reiniciar o computador e a placa de rede terá um novo endereço físico.

9.6.3. Burlando o WEP

Para realizar esse procedimento será seguido como exemplo um estudo já realizado [Souza 2005], onde o mesmo realiza um estudo de caso que demonstra na prática a quebra do WEP, com o uso de três ferramentas. São elas: Airodump, Aireplay e Aircrack. Para utilização dessas ferramentas é necessário qualquer sistema operacional de uma distribuição Linux. O primeiro passo foi conhecer qual rede será monitorada para realizar a quebra do WEP. Para conhecer a rede foi utilizado o Airodump, que exibe as informações necessárias sobre a rede. Na figura 9.19 é ilustrado isso em detalhes.

As informações necessárias são: o BSSID (endereço físico do ponto de acesso), tipo de criptografia, que no caso é o WEP e o ESSID, que é o nome de identificação da rede. Após a execução do Airodump foram conhecidos as informações necessárias da rede e posta agora a interface de rede em modo monitor, para que seja capaz de escutar o trafego das redes sem fio. O que é apresentado em detalhes na figura 9.20. Posto em modo monitor, foi necessário capturar os pacotes da rede desejada, onde é informada qual a interface de rede, qual o nome do arquivo que irá armazenar os pacotes coletados e o filtro MAC, que filtra a rede para a captura dos pacotes. Observe as figuras 9.21 e 9.22.

O mais importante do tráfego são os IVs, porque só através deles é possível a quebra da senha. Entretanto, para que o IV seja capturado, são necessários cerca de

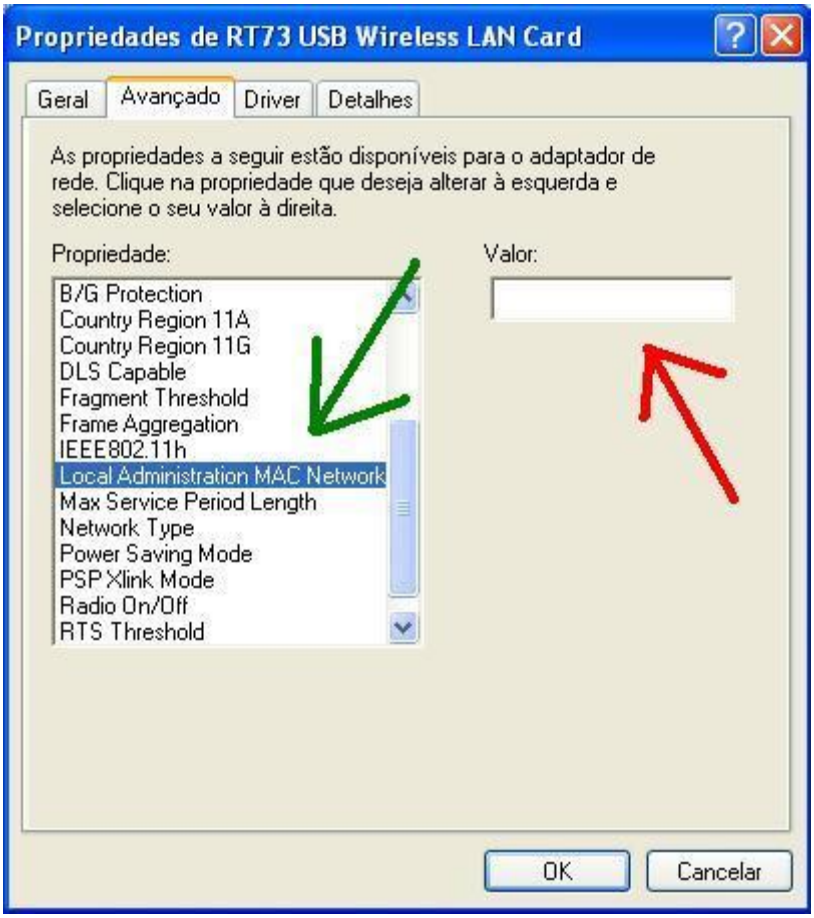


Figura 9.17. Propriedades da placa de rede

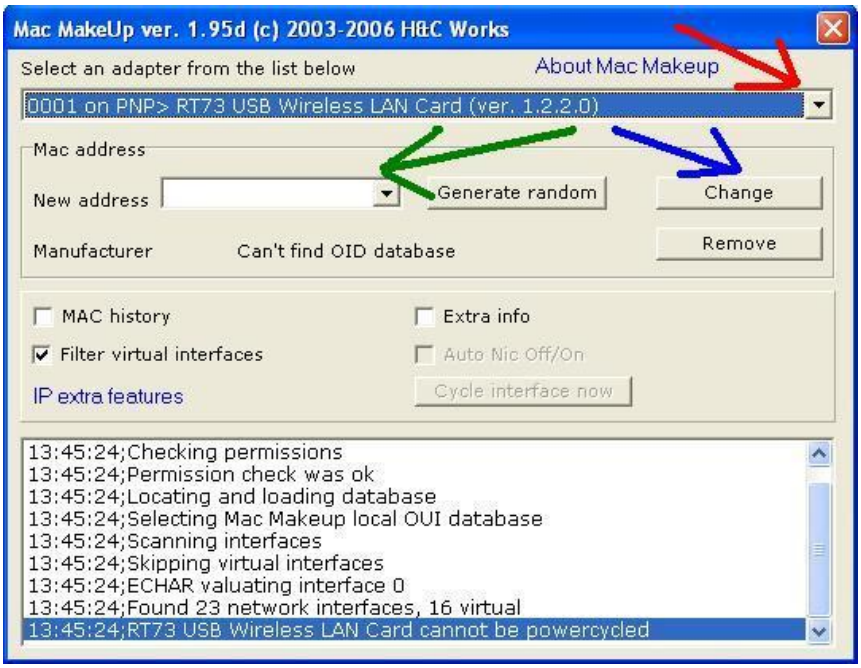


Figura 9.18. MacMakeUp em execução

Figura 9.19. Airodump

100.000 a 700.000 IVs. Porém, para a captura dessa quantidade de IVs será necessário um tráfego de rede intenso, para que a captura não dure muito tempo. Para se obter esse tráfego intenso a ferramenta Aireplay fará o serviço de inundar o ponto de acesso com pacotes do tipo ARP request, aumentado o tráfego na rede através de um cliente válido. Na figura 9.23 é ilustrado o Aireplay inundando a rede.

Após esse procedimento, em pouco tempo, ter-se-á a quantidade necessária de IVs para a tentativa de quebrar a senha. Na figura 9.24 é ilustrado o Airodump com a quantia necessária de IVs.

Com a captura necessária de IVs pelo tráfego da rede, então entra a terceira ferramenta, o Aircrak, que fará o processo para descobrir a chave WEP, com a utilização da seguinte sintaxe: aircrack -f <Fator Fudge > -q 3 <Nome do arquivo com o tráfego>. E a partir daí, o programa inicia o processo para descobrir a chave WEP. Na figura 9.25 é ilustrado o Aircrack em execução até a descoberta da chave WEP.

A partir daí, a chave é identificada, e a segurança da rede com uso de criptografia WEP foi burlada.

Figura 9.20. Colocando a Interface em modo “monitor”. Fonte: SOUZA (2005, p. 53)

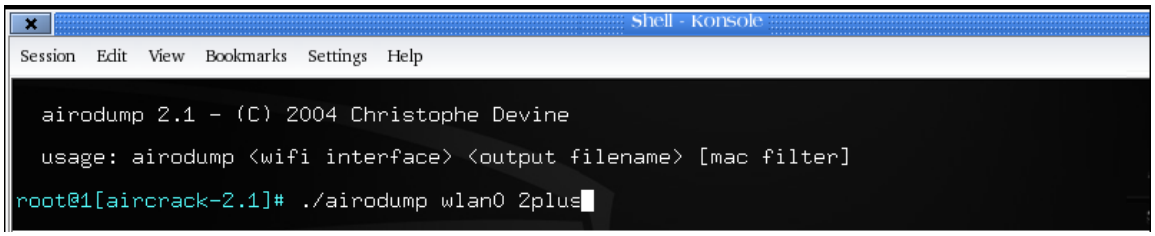


Figura 9.21. Sintaxe Airodump. Fonte: SOUZA (2005, p. 53)

9.6.4. Burlando WPA

O WPA resolveu os principais problemas do WEP, por exemplo, melhora no processamento de criptografia das chaves e aumento do vetor de inicialização para 48 bits, o que torna impossível de descobrir a chave por força bruta. O tamanho da chave em caracteres que o WPA permite é de 8 caracteres no mínimo e 63 caracteres no máximo. Assim, testar todas as possibilidades, por mais que seja uma super máquina, levaria milhões de anos com uma chave razoável com mais de 10 caracteres. Portanto, o único meio prático seria um ataque de dicionário, onde se testariam todas as palavras contidas. Porém, só teria eficácia se a palavra chave utilizada pelo usuário estivesse no arquivo (dicionário).

Esses arquivos são conhecidos como wordlist e são facilmente encontrados na Internet. COZER (2006) explica que palavras chave com menos de 20 caracteres estão sujeitas a esses tipos de ataques pelo mesmo motivo de ser comum uma chave com menos de 20 caracteres, seja ela pré-configurada pelos fabricantes ou pelos próprios administradores.

Então, para a tentativa de quebra do WPA são utilizados todos os passos e ferramentas como utilizados na quebra do WEP. Porém, este terá três diferenças simples: a primeira é a necessidade da versão mais atual do pacote Aircrack que no caso é o pacote Aircrack-ng, disponível no próprio site do Aircrack; a outra diferença está na captura dos dados com a ferramenta ainrodump-ng, onde os arquivos capturados terão que conter informações de um cliente que se autenticou com o ponto de acesso, isso porque o ataque de dicionário consiste somente em conhecer as informações de autenticação de um cliente. E por fim, será usada na sintaxe pelo Aircrack-ng, que agora segue como exemplo: aircrack-ng -e <rede> -w <dict.txt> <logrede.cap>, onde o "rede"indica o SSID da rede, o "dict.txt"indica a localização do dicionário e o "logrede.cap"é o arquivo com a captura feita pelo airodump-ng.

Indicar o SSID significa especificar qual rede será o alvo, porque é através do SSID que o cliente se autentica e portanto, essas informações terão que estar contidas no

Figura 9.22. Airodump Capturando o tráfego. Fonte: SOUZA (2005, p. 53)

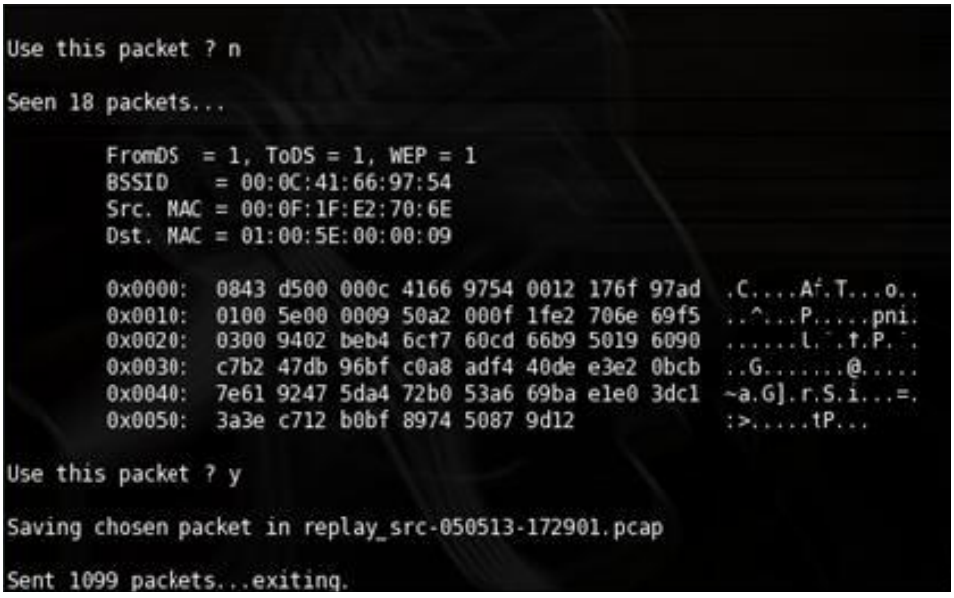


Figura 9.23. Aireplay inundando o AP de pacotes ARP. Fonte: SOUZA (2005, p. 54)

pacote que foi capturado. O ataque de dicionário só terá êxito se o administrador de rede fizer uso de uma chave que poderá estar contida no dicionário. Isso corresponde a palavras ou expressões que juntas associam uma palavra válida que existe em dicionário e que também correspondem a um número inferior a 20 caracteres, ou seja, uma senha com poucos caracteres ou de fácil adivinhação.

Caso uma chave seja bem elaborada, como o uso de letras maiúsculas, minúsculas, números e símbolos, será impossível tanto um ataque de força bruta como de dicionário decifrar esse tipo de chave.

Outro fator acaba de vez com a reputação no nível de segurança do WPA foi a comprovação que dois cientistas japoneses desenvolveram, de acordo com a Computerworld, em um evento realizado no Joint Workshop on Information Security, sediado em Taiwan. Os dois cientistas japoneses Toshihiro Ohigashi da Universidade de Hiroshima e Masakaty Morii da universidade de Kobe desenvolveram uma técnica através da qual, em questão de minutos, foram capazes de ler todo o tráfego da rede criptografada

em WPA. Essa técnica ainda não foi divulgada mas, bastou essa comprovação, para se poder dizer

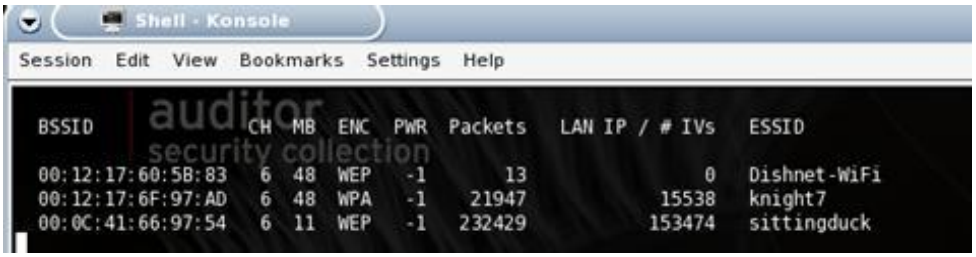


Figura 9.24. Airodump com os IVs incrementado. Fonte: SOUZA (2005, p. 54)

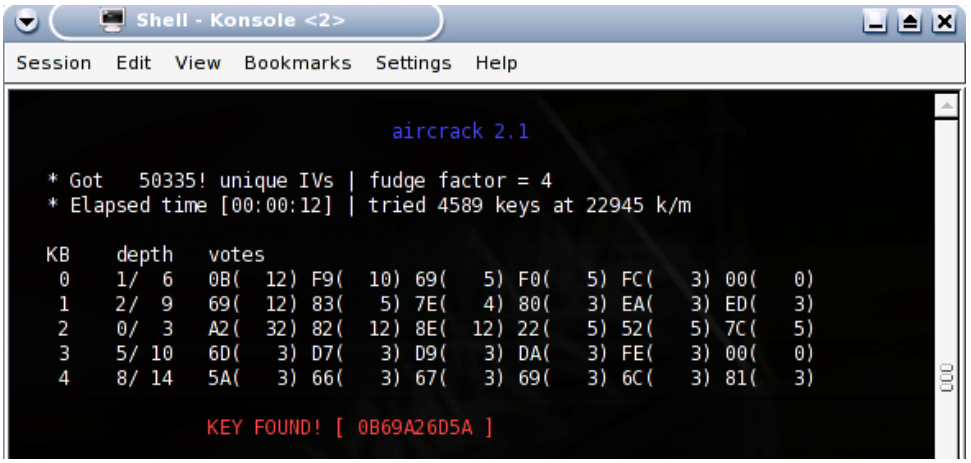


Figura 9.25. Aircrack em funcionamento. Fonte: SOUZA (2005, p. 55)

que o WPA não é mais seguro.

9.6.5. Burlando WPA2

No WPA2 não existem até o momento, trabalhos científicos ou qualquer coisa do tipo que demonstre ou prove algum tipo de falha. E isso se deve a sua principal característica, que no caso é o AES, um algoritmo muito forte de criptografia. Ainda de acordo com a Computerworld é comentando que mesmo após o conhecimento que o WPA não é mais seguro, ainda existe uma alternativa, que no caso é o WPA2, que existe desde março de 2006, porém, ainda é pouco usado nas bases instaladas pelo mundo. Este, por sua vez, é identificado como o mecanismo de criptografia mais seguro da atualidade.

9.7. Técnica dos Intrusos e os dois grandes problemas das redes WI-FI

De acordo com os estudos realizados e apresentados nas seções anteriores deste trabalho, foi comprovada a fragilidade na maioria dos mecanismos de segurança das redes WI-FI. Contudo, um mecanismo de segurança se identificou como o melhor e mais seguro da atualidade para a segurança dos dados trafegados na rede WI-FI, o WPA2. Porém, podese perceber neste trabalho, que o WPA2 e seus antecessores só protegem os dados que trafegam na rede e não a estrutura da rede como um todo. Portanto, identificaram-se dois grandes problemas que simbolizam as fragilidades na segurança das redes WI-FI. São eles: Sinal de rádio frequência propagados no ar e Clonagem do endereço MAC.

O problema das informações que se propagam no ar está referente ao tipo de informação que se faz necessário para autenticação dos clientes no ponto de acesso. No

mínimo dois textos puros são passados sem criptografia, ou seja, esses textos correspondem ao endereço MAC e o endereço IP, e são necessários para que as informações cheguem até o seu destino e não podem ser passados como textos disfarçados (criptografados). É através deles que as placas correspondentes a esses endereços fazem a comunicação com o ponto de acesso.

Os mesmos problemas também se seguem ao ponto de acesso, ou seja, um ponto de acesso também contém um endereço MAC. Os clientes se utilizam desse para fazerem a autenticação com o AP. Portanto, utilizando-se de software sniffes é possivel detectar esse endereço do ponto de acesso. E aí se leva ao problema maior, que no caso é a clonagem do endereço MAC. Este problema será descrito na subseção seguinte.

9.7.1. Sinal de rádio frequência propagado no ar

Como o sinal é propagado no ar, foi analisado que nada se pode fazer quando o intruso quer ler ou capturar o tráfego da rede. Independentemente de haver ou não serviços de encripitação, esse sinal pode ser facilmente capturado e salvo pelo intruso. Os mecanismos de encripitação só podem assegurar a questão da integridade das informações. Essas informações estão encriptadas por algoritmos que disfarçam o conteúdo real, porém, mesmo assim, correm o risco de serem desencriptadas.

9.7.2. Clonagem do endereço MAC

Como entendido no tópico anterior, que não há meios que impeçam a leitura e capturas das informações, então, não há também como impedir que eles conheçam os endereços MAC e IP da rede, tanto dos clientes como do ponto de acesso.

Conhecendo-se o endereço MAC e o IP utilizado na rede fica fácil a clonagem dos mesmos. E então os intrusos geram um grande problema, que é a técnica já conhecida como D.O.S. A técnica de D.O.S, por ser uma técnica capaz de negar algum tipo de serviço, é capaz de negar todos os serviços de uma rede. Se um intruso se fizer passar por um ponto de acesso, então todos os clientes, ou principalmente os mais próximos terão problemas nos níveis de comunicação com o ponto de acesso.

Com base nos estudos avaliados nas seções anteriores, pode-se encontrar o SSID e também clonar o MAC. Tudo isso, sem nenhuma intervenção dos meios de segurança. Contudo, se um intruso utilizar esses recursos e criar um ponto de acesso com as mesmas características do AP real da rede ocorrerá um conflito real para os clientes. Por mais que o AP use segurança de criptografia ou algum controle de acesso de clientes não será capaz de se evitar que seus clientes tenham conflitos de comunicação.

Para a comprovação dessa análise foi feito um estudo de caso como teste51 que evidenciou esses fatos. Esse teste foi realizado com os seguintes equipamentos: dois *notebooks*, um computador desktop e um roteador. O computador desktop é conectado à internet normalmente e sua conexão é compartilhada com o roteador. O roteador foi configurado com uma segurança adequada, através da utilização do WPA2. O IP foi configurado com o número 192.168.1.1, o qual servirá de ponto de acesso para os *notebooks* (clientes). Um dos *notebooks* foi conectado como cliente desse roteador. Para a cone xão com o roteador só foi informada a senha de acesso, como configurado no roteador. Detalhes são apresentados na figura 9.26.

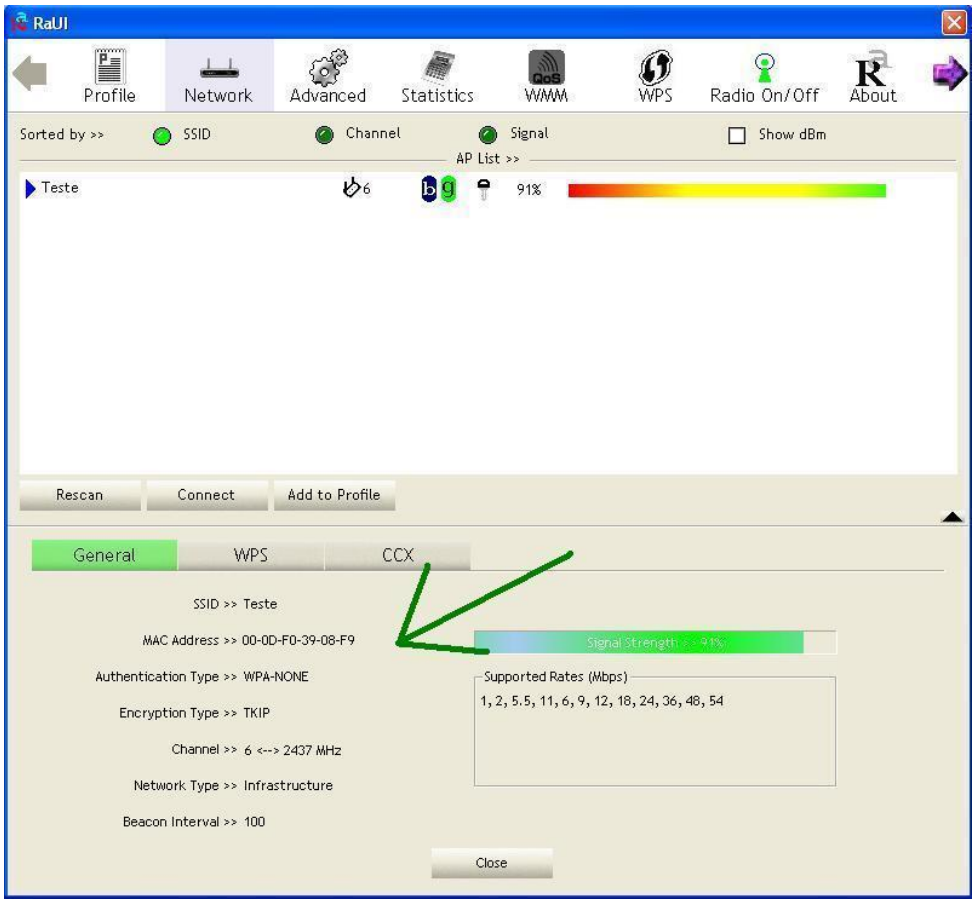


Figura 9.26. Cliente (notebook) conectado

O software utilizado para conexão com o roteador foi o do próprio fabricante, isto é, o Ralink. No software (indicado com seta verde) mostram-se as propriedades do ponto de acesso na qual o notebook está conectado. A seguir são descritas as principais informações do AP:

- SSID da rede identificado como Teste;
- MAC Address que corresponde ao MAC do ponto de acesso, com o número 000DF03908F9;
- Autenhtication Tupe que significa o tipo de autenticação utilizada, neste caso WPA/TKIP;
- channel que é o número do canal utilizado pelo AP. A posse dessas informações é suficiente para que através do outro notebook sejA criado um ponto de acesso idêntico ao AP real. O AP foi criado da seguinte forma: foi clonado o MAC do AP, como os métodos explicados anteriormente. O SSID foi denotado como o mesmo do AP, no caso, o nome “Teste”. E por último foi escolhido o mesmo canal do AP, o canal de número 6.

Como ilustrado na figura 9.27, foi utilizado o software Ralink para a criação do ponto de acesso. Nela a seta azul indica que foi posto em modo AP. A seta vermelha indica o nome do SSID, com o nome “Teste”. E por fim, a seta verde indicando o canal, no caso o número 6. Para finalizar, foi configurado na placa de rede, que está sendo posta como ponto de acesso, o mesmo IP do roteador, ou seja, o IP 192.168.1.1., resultando em duas redes idênticas para o acesso.

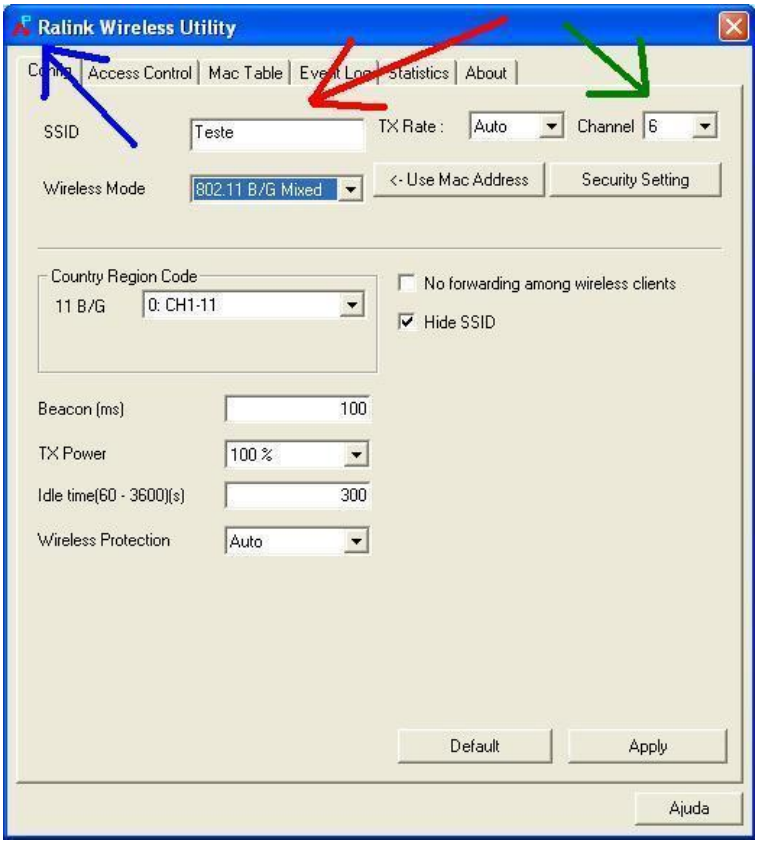


Figura 9.27. Ponto de acesso criado

O problema é evidenciado da seguinte forma: no momento que foi criado o ponto de acesso clone, com as mesmas características do AP, o cliente (o outro notebook que estava conectado ao roteador), se desautenticou e autenticou novamente, ou seja, a rede caiu e se restabeleceu. Isso se deu pelo fato de o ponto de acesso clone ter informado ao cliente serviços idênticos ao do roteador.

Analisando esse problema a fundo: se isso ocorresse em uma rede com muitos clientes conectados, a rede ficaria caótica com tantos clientes se desautenticando e tentando se autenticar novamente.

9.7.3. Ocorrências de intrusos

Com base nos estudos realizados e apresentados neste capítulo, detectar a ocorrência de intrusos é algo que tem muito que evoluir. Mesmo assim foram detectados que dois fatores são cruciais para poder identificar um intruso na rede. São eles:

Aumento do tráfego na rede: O aumento do tráfego de rede pode ser algo de uma situação maliciosa, principalmente quando a rede utiliza-se de criptografia, ou seja, o intruso utiliza técnicas capazes de aumentar o tráfego de rede, introduzindo pacotes de requisição de serviços ao roteador. Aumentando esse tráfego, fica mais fácil capturar a quantidade desejada de pacotes e com isso utilizar-se de técnicas, para conhecer as informações contidas nos pacotes através da quebra de criptografia.

Problemas na conexão dos clientes: Quando são notados problemas nas conexões dos clientes, ou até mesmo nos níveis de comunicação e autenticação, é muito provável

que a rede esteja sendo atacada. Isso se deve ao ataque de negações de serviços, que então, de alguma maneira, o intruso está tentando se passar por um cliente ou ponto de acesso.

9.8. Considerações Finais

O desenvolvimento deste trabalho teve como base o conhecimento da estrutura das redes WI-Fi e suas vulnerabilidades decorrentes dos níveis de segurança oferecidos. Através desse estudo, foi verificado que todos os meios de segurança, exceto o WPA2, possuem fragilidades comprovadas. Algumas medidas de segurança foram estudadas para prover o conhecimento necessário de como os intrusos fazem uso das fragilidades existentes nas redes sem fio. A maior causa dos ataques hoje é devido a essas redes estarem mal configuradas ou sem nenhuma configuração, ou ainda com a configuração de fábrica. Os softwares utilizados por atacantes são facilmente encontrados na web, o que aumenta a insegurança em redes sem fio. Há ainda muito que evoluir na segurança em relação ao gerenciamento da rede WI-FI, portanto a utilização de redes sem fio é segura desde que utilizada com equipamentos modernos com melhoria de seguranças e junto com métodos de autenticação. Outro fator importante deste estudo foi à nítida preocupação dos fabricantes em aumentar a segurança, bem como a velocidade de seus equipamentos de redes sem fio, como foi o adiantamento do protocolo WPA, substituindo o protocolo WEP, como medida de urgência em seus equipamentos.

9.9. Referências

[Albuquerque 1999] Albuquerque, F. (1999). TCP/IP – Internet: Protocolos & Tecnologias. Axcel Books do Brasil Editora, Rio de Janeiro. 2aed.

[Alliance 2012] Alliance, W. (2012). Wifi certifiedmakes it wi-fi. Wi-Fi Alliance. Disponível em: http://www.wi-fi.org/.

[Andrews et al. 2007] Andrews, J. G., Ghosh, A., and Muhamed, R. (2007). Fundamentals of WiMAX: Understanding Broadband *Wireless* Networking. Prentice Hall Communications Engineering and Emerging Technologies Series, Upper Saddle River, NJ, USA.

[Calazans et al. 2001] Calazans, N. L. V., Moraes, F. G., Torok, D. L., and Andreoli, A. V. (2001). Projeto para prototipação de um ip soft core mac ethernet. Revista RITA. Valume VIII. Número 1.

[Comer 2006] Comer, D. E. (2006). Interligação em Redes com TCP/IP, volume 1 Princípios, protocolos e arquitetura. Campus, trad. 5 ed. edition.

[Castro Peixoto] de Castro Peixoto, R. Tecnologias *wireless* demandam cuidados extras a prática do *Wardriving* e *Warchalking*. Disponível em: http://www.correiadasilva.com.br/pdf/art_dig/art_dig11.pdf.

[de Morais Jardim 2007] de Morais Jardim, F. (2007). Treinamento Avançado em Redes *Wireless*. Digerati Books, São Paulo.

[Dhar] Dhar, S. *Sniffers* Basics and Detection. Versão 1.0-1. Disponível em: http://www.linux-sec.org/Sniffer.Detectors/*Sniffers*.pdf.

[Dias and Jr. 2002] Dias, B. Z. and Jr., N. A. (2002). Evolução do padrão ethernet. CBPFNT-002/02. Disponível em: mesonpi.cat.cbpf.br/naj/ethernet.pdf.

[Held 1999] Held, G. (1999). Comunicação de dados. Editora Campus, Rio de Janeiro. 6ª ed.

[IEEE 2012] IEEE (2012). Ieee institute of electrical and electronics engineers advancing technology for humanity. IEEE, Org. Disponível em: http://www.ieee.org/.

[JiWire 2012] JiWire (2012). Top 10 countries rank, top 10 cities rank. JiWire, Inc. Disponível em: http://wi-fi.jiwire.com/.

[Kurose e Ross 2006] Kurose, J. F. and Ross, K. W. (2006). Redes de computadores e a Internet uma abordagem top-down. Pearson Addison Wesley, São Paulo. 3 ed.

[Peterson and Davie 2004] Peterson, L. L. and Davie, B. S. (2004). Redes de Computadores: Uma abordagem de sistemas. Campus, Rio de Janeiro, trad. 3 ed. edition.

[Prodanovic and Simic 2007] Prodanovic, R. and Simic, D. (2007). A survey of *wireless* security. Journal of Computing and Information Technology CIT 15, page 237–255.

[Soares et al. 1995]Soares, L. F. G., Lemos, G., and Colcher, S. (1995). Redes de Computadores. Das LANs MANs e WANs às Redes ATM. Editora Campus, Rio de Janeiro. 2a ed.

[Souza 2005] Souza, R. M. (2005). Análise das vulnerabilidades e ataques existentes em redes sem fio. Mografia.

[Tanenbaum 2011] Tanenbaum, A. S. (2011). Redes de Computadores. Pearson, São Paulo. Trad. 5 ed.

[Torres 2001] Torres, G. (2001). Redes de Computadores Curso Completo. Axcel a Books do Brasil Editora, Rio de Janeiro. 1 ed.

[Whalen 2002] Whalen, S. (2002). Analysis of WEP and RC4 Algorithms. Disponível em: http://www.rootsecure.net/content/downloads/pdf/wep_analysis.pdf.

Capítulo
10

Projeto de interface de usuário em aplicações móveis: uma introdução à plataforma Android

João Soares de Oliveira Neto, Ernando Passos

Resumo

Os profissionais que trabalham com desenvolvimento de software devem estar preparados para atender a uma das maiores expectativas no mercado tecnológico atual: a demanda por aplicações que serão executadas em dispositivos móveis. Este capítulo discute assuntos que devem ser vistos como fundamentais a fim de se diferenciar no mercado já competitivo. O primeiro assunto é a Interação Móvel, ou seja, como o usuário dialoga com as aplicações no contexto onde a mobilidade é primordial. Em seguida, são apresentados os fundamentos da plataforma Androide, sua arquitetura, os componentes de uma aplicação, os elementos da interface de usuário e alguns tópicos avançados, como a conexão com banco de dados e o uso da Google Maps API.

10.1. Introdução

A mobilidade é uma das características marcantes deste início de século. Os benefícios trazidos pela popularização da Computação nas últimas décadas do século passado já se mostravam restritivos, nessa área que se reinventa em passos curtos. O surgimento da Computação Móvel (*Mobile Computing*) transforma a maneira como o indivíduo e as instituições contemporâneas se relacionam com os outros, como realiza transações financeiras, como acessa serviços e informações e torna possível o cenário projetado por Marc Weiser, em que o indivíduo permanece conectado à Internet o tempo todo e de (todo e) qualquer lugar (Weiser 1991).

Os fornecedores de software começaram a desenvolver aplicações móveis (ou, simplesmente, *apps*) para as mais variadas finalidades – jogos, localização de lugares e de pessoas, redes sociais, transações financeiras, entre outras. A conexão sem fio, o tamanho reduzido e o peso leve dos equipamentos, e a libertação dos fios/cabos produziram um território completamente novo para a Computação se expandir. Assim, os dispositivos móveis tornaram-se acessórios fundamentais.

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

A preocupação com a infraestrutura de comunicação e com a invenção de novos recursos para adicionar aos equipamentos móveis, deixaram para segundo plano questões ligadas à interação do usuário com estes dispositivos. Ou seja, à interação do usuário com a interface específica para dispositivos móveis. De certa forma, isto se justifica pelo fato de que, no início, as *apps* eram nada mais do que miniaturização ou adaptações de serviços/aplicações projetadas inicialmente para o desktop (Yamabe *et al.*, 2008).

A disciplina de IHC (interação humano-computador), que investiga a relação entre usuários, sistemas computacionais e aplicações, abriu espaço para a Interação Móvel. Métodos, técnicas e ferramentas começaram a ser criados para a sistematização do estudo da maneira como o usuário interage com dispositivos móveis: as limitações, necessidades, e como tornar este diálogo mais fácil para o usuário. O contexto de uso torna-se muito importante, pois, com a mobilidade, o usuário sai do ambiente quase que controlado e até determinístico do escritório, da sala de estar, do balcão de um bar, entre outros. Se a máxima de que o usuário é peça fundamental para a garantia da qualidade de uma aplicação, na Interação Móvel, o contexto em que o usuário se encontra não pode ser menosprezado, pois ele interfere diretamente na forma como o indivíduo utiliza os sistemas computacionais e as aplicações.

O objetivo deste capítulo é apresentar aos profissionais envolvidos no desenvolvimento de aplicações para dispositivos móveis – sobretudo as áreas de IHC e Engenharia de Software – os princípios de diretrizes gerais para o projeto de interfaces de usuários para dispositivos móveis. Além disso, são discutidas as especificidades da Interação Móveis, ou seja, as características que torna esse paradigma diferente dos paradigmas tradicionais de IHC. Além disso, é mostrada a coleção de componentes gráficos disponíveis na plataforma Android, que tem ganhado cada vez mais atenção de pesquisadores e profissionais por ser gratuita, originalmente idealizada para dispositivos móveis, robusta, integrável a uma série de serviços e APIs amplamente conhecidas, e conta com uma ampla (e crescente) variedade de equipamentos capazes de executar suas aplicações.

Este capítulo está estruturado da seguinte forma: na Seção 2 são apresentados e discutidos os princípios da Interação Móvel e são discutidos os princípios de projeto para que o usuário tenha uma experiência positiva ao usar aplicações desenvolvidas para aplicações móveis; a Seção 3 concentra-se na plataforma Android, apresentando sua arquitetura, os componentes de uma aplicação Android, os principais componentes de interface de usuário para entrada e saída de dados, e aborda alguns tópicos avançados, tais como a conexão a bancos de dados e o uso de classes e interfaces da *Google Maps API*; e, finalmente, a Conclusão, na Seção 4.

10.2. Interação Móvel

A mobilidade é uma das características marcantes deste início de século. Os benefícios trazidos pela popularização da Computação nas últimas décadas do século passado já se mostravam restritivos, nessa área que se reinventa em passos curtos. O surgimento da Computação Móvel (*Mobile Computing*) transforma a maneira como o indivíduo e as

instituições contemporâneas se relacionam com os outros, como realiza transações financeiras, como acessa serviços e informações e torna possível o cenário projetado por Marc Weiser, em que o indivíduo permanece conectado à Internet o tempo todo e de (todo e) qualquer lugar (Weiser 1991).

Para B'Far (2005), o projeto de interface de qualquer aplicação para dispositivos móveis deve preocupar-se com: a aparência e o comportamento da aplicação e como torná-la agradável aos usuários; a facilidade da aprendizagem da interface e a sua eficiência; e questões de saúde no uso da interface.

A maioria dos dispositivos móveis ainda possuem telas pequenas, o que interfere na quantidade de informações que pode ser apresentada ao usuário num dado momento. Não é fácil projetar uma interface de usuário, como um guia de turismo, que fornece informações sobre cinemas, restaurantes, museus, entre outros, e possibilite que o usuário encontre a informação que ele precisa, sem que este seja um processo com vários níveis e camadas de dados. Os usuários com dificuldades de localização espacial sentirão muita dificuldade em utilizar um sistema com uma interface que exija a navegação por vários níveis abaixo e que o usuário volte todo o caminho percorrido se quiser acessar outra seção do site.

Além da confusão espacial, deve-se lembrar de que as pessoas usam os aparelhos celulares e suas *apps* em ambientes sociais e dinâmicos, onde vários elementos podem contribuir a distraí-las enquanto tentam acessar as informações e serviços. Entre estes elementos, podem ser citados: barulho do trânsito, outras pessoas andando distraídas na rua e esbarrando em outras pessoas, muita/pouca luz ambiente etc. Além disso, outro fator do contexto muito importante é a qualidade da conexão à rede de dados. Uma conexão fraca pode fazer com que o usuário passe muito tempo tentando encontrar um restaurante onde fazer uma refeição.

De acordo com a norma ISO 13407, detalha o Processo Centrado no Usuário na criação de sistemas interativos, são quatro os pontos fundamentais a serem observados na criação de uma aplicação interativa:

- Entender e especificar o contexto de uso;
- Especificar os requisitos do usuário e da organização;
- Produzir protótipos;
- Conduzir avaliações com usuários reais.

O usuário pode necessitar usar a aplicação quando está se deslocando numa avenida movimentada, dentro de um ônibus lotado e em movimento, ou ainda, tendo alguém do lado tentando estabelecer uma conversa. Ou seja, a atenção do usuário no contexto móvel é disputada a maior parte do tempo. Este princípio determina que as *apps* devem ser objetivas, diretas e com poucas páginas. Nielsen (2011), na Figura 10.1, apresenta um contraexemplo deste princípio. Neste contraexemplo, a página possui muito texto, e uma fonte muito pequena para a maior parte do texto, que dificilmente será lido na sua

totalidade pelo usuário em momentos de correria. O desenvolvedor deve prover uma aplicação que funcione em qualquer contexto.

Figura 10.15 - Contraexemplo de uma página direta
Fonte: (Nielsen, 2011)

A versão *mobile* da Wikipedia é um exemplo positivo do uso do princípio da objetividade. Como pode ser observado na Figura 10.2, ao mostrar o resultado de uma busca do usuário, a página traz a definição, seguida de várias categorias que poderão ser detalhadas se o usuário tiver o interesse, e clicar sobre cada categoria.

A entrada de dados é outro quesito ainda crítico na interação com dispositivos móveis. Cada produto apresenta uma solução para este problema, dependendo da sua capacidade de processamento, tamanho do dispositivo e recursos adicionais, tais como teclas ou tela sensível ao toque, e da aplicação em si. O uso de caneta *stylus*, utilizada há algum tempo, facilita a entrada de dados em dispositivos com telas pequenas, enquanto que o contato direto dos dedos, mais recente, pode ser muito útil em telas maiores para elaborar desenhos ou para montar palavras deslizando o dedo sobre as letras do teclado, em vez de "digitar" sobre teclas uma-a-uma.

Para facilitar ainda mais a inserção de dados, tem-se observado um crescimento no de interação por voz, o que, além de tornar natural o diálogo com os dispositivos móveis, pode deixar as mãos livres, por exemplo, para realizar outras tarefas, e facilitar o uso por pessoas que tenham dificuldades motoras. Outra tendência é a mistura, ou associação, de modalidades, por exemplo, uso de voz na inserção de dados e uso do teclado virtual para corrigir eventuais erros no processo de captura de comandos falados.

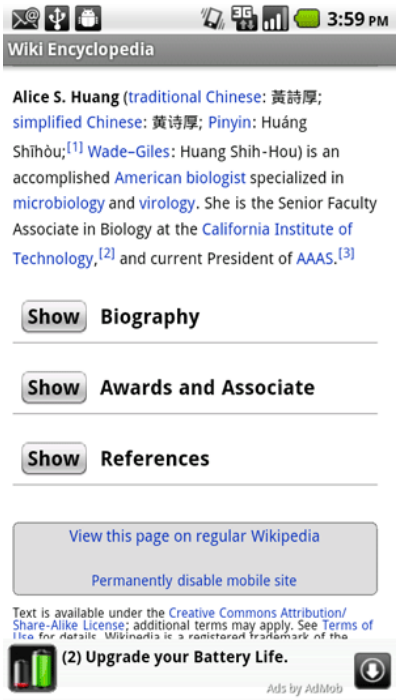


Figura 10.16 – Exemplo positivo de uma página móbile objetiva
Fonte: (Nielsen, 2011)

Outro exemplo a ser seguido aparece na Figura 10.3. À esquerda, aparece a primeira página da *apps*, se o usuário precisar de mais informações sobre uma categoria, basta clicar no link correspondente, que aparecerá a página seguinte, que está à direita da figura. Para Nielsen (2011), a regra é simples: "Se você fizer a primeira página muito densa, ninguém vai ler nada. Melhor mostrar na primeira página a essência do seu conteúdo. Assim, somente os usuários realmente interessado em se aprofundar neste conteúdo irá prosseguir para as páginas seguintes. Isso poderá agregar valor ao seu conteúdo".

Os sensores embarcados em boa parte dos dispositivos móveis possibilita ao usuário uma forma avançada de entrada de dados. Graças aos sensores é que se pode capturar movimentos, e gestos como o toque com um ou mais dedos – como o gesto de pinça.

B'Far(2005) lembra ainda que as *apps* devem ter interface consistente com eventuais versões existentes para outras plataformas, tais como *desktop* e TV Digital. O autor chama atenção também para o fato de que a aplicação deve respeitar as limitações do usuário e não sobrecarregá-lo com respostas em excesso e sobrepostas. A respeito da infraestrutura, não se pode esquecer que os equipamentos ainda possuem autonomia de bateria muito pequena ainda e que a velocidade das conexões é baixa. Para o projetista de interface isso deve ser traduzido em transações rápidas e que não desperdicem recursos computacionais – como uso do processador e da memória.

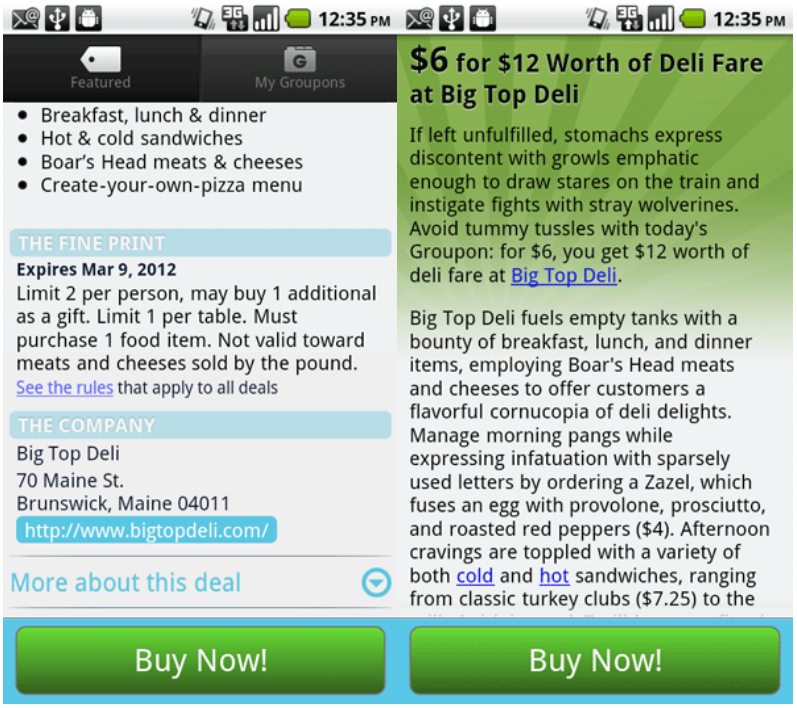


Figura 10.17 - Exemplo positivo da divisão de conteúdo entre a primeira e as demais páginas
Fonte: (Nielsen, 2011)

A entrada de dados requer mesmo uma atenção maior dos desenvolvedores. Em aplicações baseadas em toque na tela, a área sensível ao toque – a região que dispara alguma ação ao ser tocada – deve ser grande o suficiente e distinta das outras áreas. Condey e Darcey (2001) recomendam que o tamanho da fonte e dos ícones seja grande também. Deve-se reduzir ao máximo os cliques e a digitação. E, por outro lado, não se pode confiar num mecanismo particular de entrada de dados estará disponível para todos os dispositivos – uma tecla específica ou presumir a existência de um teclado.

Em resumo, o usuário (i) está em movimento, ou na iminência de se movimentar, entre localidades previamente conhecidas ou não; (ii) tipicamente não está concentrado em uma atividade somente; (iii) espera um alto grau de resposta e imediatismo do sistema; (iv) muda frequentemente de atividade, e de maneira abrupta; e (v) necessita da *apps* em qualquer lugar a qualquer momento.

10.3. A plataforma Android

10.3.1. Surgimento

O crescimento e a popularização da Web propiciou um grande desenvolvimento de tecnologias para comunicação por meio de dispositivos móveis. Tal evolução desencadeou

uma maior oferta de aparelhos contendo Sistemas Operacionais mais complexos no mercado, fazendo com que a população adquirisse cada vez com mais frequência os dispositivos móveis, transformando-os assim em uma realidade no nosso cotidiano (Mack, 2011). Como consequência, além de obtermos uma maior variedade de *tablets* e *smartphones* no mercado, os usuários foram contemplados com o surgimento de diversos Sistemas Operacionais para os aparelhos. Surgiu então uma grande dificuldade para criação e difusão de aplicativos, pois cada empresa tinha o seu Sistema Operacional padrão, e havia incompatibilidades entre seus sistemas, ou seja, os aplicativos em sua maioria eram limitados apenas aos aparelhos das mesmas empresas. Logo houve a necessidade da criação de uma plataforma única para celulares, a fim de uma maior aceitação dos consumidores.

Um grupo formado por grandes empresas no mercado de dispositivos móveis dentre elas Samsung, Sony Ericsson, Toshiba, T-Mobile, China Mobile, LG, Intel, ZTE, liderados pela Google a *Open Handset Alliance* (OHA) definiram uma plataforma única, aberta e bastante flexível para o desenvolvimento de aplicações, surgia então o Sistema Operacional Android (OHA, 2011). A lista completa com todas as empresas que fazem parte da OHA pode ser encontrada em OHA (2011).

O Android é um sistema operacional baseado em Linux que oferece como grande vantagem o fato de ser livre e de código aberto. Isto significa que cada empresa poderá customizar e modificar o sistema para a sua necessidade, sem a obrigação de compartilhar essas alterações com ninguém (Lecheta, 2010).

10.3.2. Arquitetura

A arquitetura do Sistema Operacional Android é formada por camadas, como a Figura 10.4 ilustra. A seguir é descrito o papel de cada uma delas.

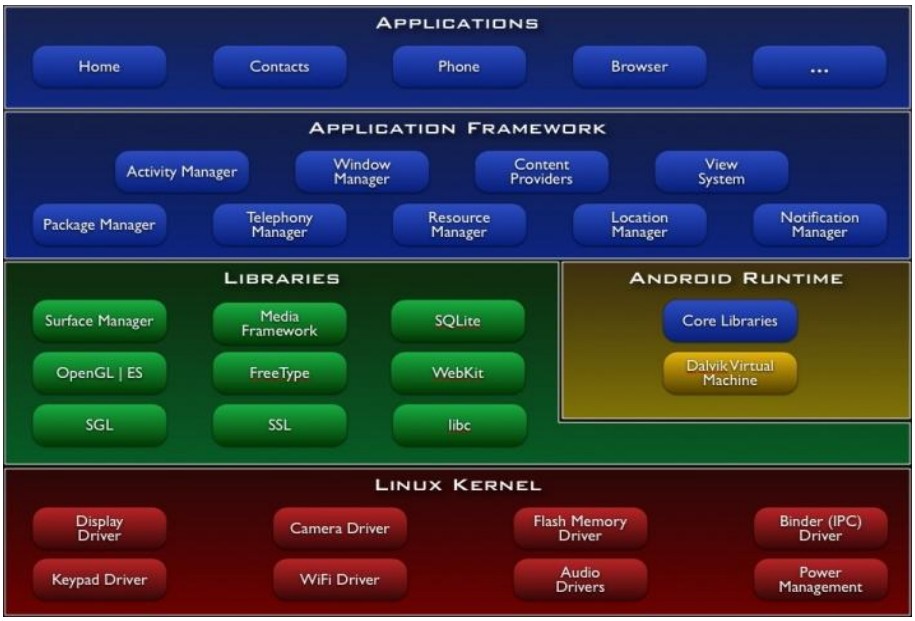


Figura 10.65 - Arquitetura Google Android
Fonte: (GOOGLE, 2011)

A arquitetura tem na sua base o *Kernel Linux* que oferece uma camada de abstração de hardware, configurações de seguranças, e funções básicas, como gerenciamento de memória, energia e dos *drivers* do dispositivo. Desta maneira, o *Kernel Linux* proporciona toda segurança, estabilidade, e portabilidade do Sistema Operacional Linux, que apesar de na visão do usuário não haver um contato direto com o Linux, para desenvolvedores durante a criação de algumas aplicações é preciso fazer o uso do *shell* (Burnette, 2010).

Logo acima do *Kernel Linux* estão as bibliotecas nativas do Android, que são criadas em C/C++, compiladas para o hardware específico do aparelho, e geralmente pré-instalada pela empresa fornecedora. As bibliotecas são responsáveis por gerenciar banco de dados, reprodução de áudio, aceleradores gráficos, entre outros. Os principais componentes da camada *Libraries* são (GOOGLE, 2011):

- **Surface Manager:** gerencia o acesso ao subsistema de exibição e as múltiplas camadas de aplicações 2D e 3D;
- **Media Framework:** bibliotecas que suportam os mais populares formatos de aúdio, vídeo e imagens;
- **WebKit:** renderizador de código aberto utilizado em navegadores;
- **SQLite:** banco de dados relacional disponível para todas aplicações.
- **OpenGL/ES:** implementação baseada no OpenGL 1.0, as bibliotecas utilizam aceleração 3D.

O *Android Runtime* é responsável pelo que ocorre em tempo de execução, e tem em sua composição a Máquina Virtual *Dalvik*, que é específica para dispositivos móveis fornecidos com o Android.

O código é feito na linguagem Java, para a criação do arquivo `.class`, que posteriormente é compilado e convertido para o formato .dex (*Dalvik Executable*). No passo seguinte, são compactados os executáveis com imagens para a geração do arquivo final, com extensão .apk (*Android Package File*). Finalmente, o arquivo está pronto para o uso no aparelho. O processo de geração do arquivo final é mostrado na Figura 10.5.

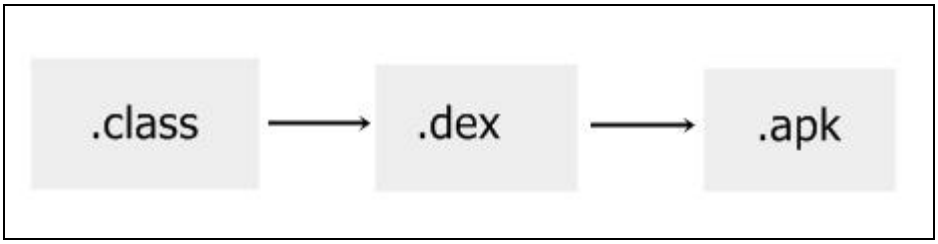


Figura 10.19 - Processo de geração do arquivo .apk

É importante salientar que com o recurso do SDK do Android e a utilização do ambiente de desenvolvimento Eclipse, todas essas etapas são feitas automaticamente (Lecheta, 2010).

A camada *Applications Framework* tem a função de administrar as funções básicas dos aplicativos no Android, tais como gerenciamento de contatos, alarmes, barra de *status*, *Activity Manager* - responsável pelo gerenciamento do ciclo de vida de uma aplicação -, *Location Manager* - útil para desenvolvimento em mapas, localização e GPS, além do acesso aos provedores de conteúdos do Android. A seguir são mostrados os principais componentes da camada *Applications Framework* (GOOGLE, 2011):

- **View Manager:** responsável pela construção dos componentes básicos de interface. O componente *View* desenha e manipula os eventos na tela;
- **Location Manager:** fornece acesso aos serviços de localização do sistema, posteriormente, neste capítulo, será abordado o *Location Manager* com mais detalhes;
- **Activity Manager:** gerencia o ciclo de vida das aplicações, e proporciona uma navegação em forma de pilha de *Activities*;
- **Notification Manager:** permite que as aplicações exibam notificações quando algum evento acontece, por exemplo, uma mensagem de texto ou uma ligação telefônica recebida;
- **Content Providers:** permitem que aplicações compartilhem e acessem informações de outros aplicativos;
- **Telephony Manager: f**ornece opções sobre os serviços de telefonia do dispositivo.

No topo da pilha, encontram-se as aplicações utilizadas pelo usuário tais como navegadores, cliente de e-mail, calendário, mapas, além de funções de telefonia. Tais aplicativos, desenvolvidos por terceiros, formam a camada *Applications*, que tem como característica a interação entre usuários e interface dos aplicativos. Através do *Android Market*, é possível ter acesso e instalar às informações de aplicativos para Android disponibilizados gratuitamente ou pagas por desenvolvedores ou empresas.

10.3.3. Componentes de uma aplicação Android

Uma aplicação Android é composta por:

- ***Activity* (Atividade):** tem como características o seu ciclo de vida e a responsabilidade de controlar os eventos da tela;
- ***Service* (Serviço):** geralmente, é utilizado para executar serviços em segundo plano vinculados a processos;
- ***Intent* (Intenção):** levam em sua instância informações importantes como, por exemplo, para o inicio de uma nova *Activity*;
- ***Content Provider*:** responsável por tornar públicas as informações dos aplicativos. Estes componentes são em sua maioria iniciados através de mensagens assíncronas chamadas de *Intent*.

As próximas seções detalham estes componentes e apresentam maiores detalhes do seu funcionamento.

10.3.4. Activity

Uma Activity geralmente está relacionada a uma tela da aplicação. A partir delas é possível fazer uma representação de uma interface visual e tratamento de eventos, tais como a seleção em menus ou toque na tela. Uma aplicação pode ser composta de uma composição de *Activities*, podendo haver uma alternância entre as mesmas. Cada *Activity* tem um papel específico na solução de um problema durante o processo do aplicativo. Geralmente a alternância entre *Activities* está associada à mudança de telas, isto é feito quando se envia ao Sistema Operacional uma mensagem (*Intent*), com argumentos necessários para a inicialização da tela (Lecheta, 2010).

Por exemplo, um jogo eletrônico simples poderia ter as seguintes *Activities*: uma tela de início ou boas-vindas (*Startup* ou *Splash*); um tela com o menu principal, que atua como o *Activity* principal e é o ponto central para a navegação; tela da partida, onde o jogo acontece; tela de maiores pontuações, que mostra a pontuação de todos os jogadores; e tela de ajuda/sobre, que apresenta informações que os usuários precisam para jogar.

Toda *Activity* possui um ciclo de vida que é gerenciado pelo Sistema Operacional. No momento da criação de um aplicativo é necessário levar em consideração fatores como o seu encerramento brusco por uma ligação telefônica recebida, por uma mensagem privada ou ainda, pelo recebimento de mensagens enviadas pelo Sistema Operacional. Entender o funcionamento e os possíveis estados de uma *Activity* se torna imprescindível para o desenvolvedor. O conceito de ciclo de vida está relacionado à pilha de *Activities*, ou seja, uma aplicação real pode ser interrompida por um *browser* ou uma ligação telefônica, logo a nova *Activity* será inserida no topo da pilha e a aplicação anteriormente em execução fica abaixo na pilha, passando a ser considerada em segundo plano para o Sistema Operacional, que poderá destruí-la e removê-la da pilha a qualquer momento, encerrando o seu processo com e liberando recursos.

O Android permite monitorar os possíveis estados de uma Activity através métodos herdados da classe `android.app.Activity`. São eles: `onCreate()`, `onStart()`, `onRestart()`, `onResume()`, `onPause()`, `onDestroy()`, e `onStop()`. A responsabilidade de cada método é (Condey; Darcey, 2001):

- **onCreate():** obrigatório para qualquer aplicação Android, é executado somente uma vez em todo ciclo de vida, nele indicamos o parâmetro necessário para a primeira tela do aplicativo, automaticamente invoca o método *onStart();*
- **onStart():** prepara a *view* - componente de visualização - para ser desenhada na tela e automaticamente invoca o método *onResume();*
- **onResume():** sempre é invocado após o método *onStart()*, quando a *Activity* está no topo da pilha, em plena execução, neste método há total interação entre usuário e aplicativo;

- **onPause():** chamado quando ocorre algum interceptação de eventos, ou seja, quando outro aplicativo é inserido colocado no topo da pilha, é usado geralmente para salvar os dados do aplicativo e evitar perdas devido a interrupção, caso a *Activity* volte a ser executada o método *onResume()* é novamente invocado;
- **onStop() e onRestart():** se o Android decidir encerrar a *Activity*, o método *onStop*() é chamado e a *Activity* não estará mais visível, ainda assim poderá voltar a execução através do metodo *onRestart()* reiniciando a Activity em questão;
- **onDestroy():** encerra a *Activity*, pode ser chamado automaticamente pelo Sistema Operacional ou pela aplicação através do método *finish()*.

A Figura 10.6 esquematiza a relação entre esses métodos e sequência em que eles são executados.

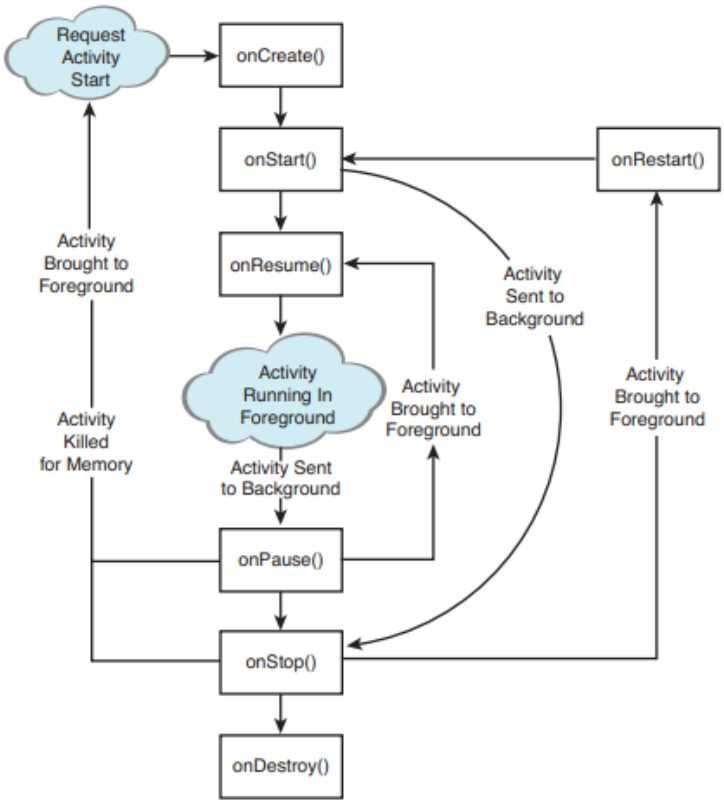


Figura 10.20 - Esquema com a relação entre os métodos de uma Activitie
Fonte: (Condey; Darcey, 2001)

Compreender o comportamento do ciclo de vida de uma Activity e os seus possíveis estados, assim como saber manuseá-los da maneira correta se torna peça chave para a garantia de um aplicativo confiável, que evite perda de informações essenciais.

10.3.5. Service

Representado pela classe *android.app.Service*, o componente *Service* tem a função de executar processos em segundo plano, em sua maioria por tempo indeterminado sem que o usuário perceba. Em um *Service* pode acontecer interação ou não com o usuário, em sua

grande maioria não há necessidade de interface gráfica a ser implementada (Burnette, 2010).

Os *Services* geralmente consomem alto recurso de memória e CPU. O seu ciclo de vida é controlado pelo Sistema Operacional, mantendo assim a integridade da execução, somente sendo encerrado quando o Android precisa liberar recursos para o seu funcionamento. Em tese, uma *Thread* faria papel similar ao *Service*, porém a *Thread* poderia ser encerrada a qualquer momento pelo Sistema Operacional o que não acontece com o serviço, que tem como característica ser controlado pelo Android (Lecheta, 2010).

10.3.6. Content Providers

O Android oferece para os programadores algumas formas de armazenamento de informações tais como banco de dados, arquivos, e o sistema de preferências. Porém quando é necessário fazer uso de dados contidos em aplicações distintas, nenhuma das formas acima citadas se torna capaz de resolver o problema, pois no modelo de segurança do Android cada aplicação tem o seu próprio diretório. Logo para que haja uma comunicação entre aplicativos foi criado o *Content Provider* (Provedor de Conteúdos) (Burnette, 2010).

O *Content Provider* é um conjunto de dados encontrados na API do Android representada pela classe (*android.content.ContentProvider*). Tem como finalidade tornar públicas informações para que todas as outras aplicações instaladas no Android possam fazer uso dos seus dados. Como exemplos de *Content Providers* nativos do Android destaca-se o acesso aos contatos do celular, a visualização de arquivos, podendo eles serem acessados a partir de qualquer aplicativo (GOOGLE, 2011).

10.3.7. Intent

Comumente na utilização de aplicativos é necessário fazer uso de algumas funções fornecidas por aplicações instaladas no aparelho ou pelo próprio Sistema Operacional. Dentre elas, podem-se citar: consulta a agenda, ligação para um contato da agenda, execução de algum processo em segundo plano fazendo uso do *Service*, execução do *browser*. Para que seja possível realizar todas estas funções, no SDK[7] do Android encontra-se a *Intent*, representada pela classe *android.content.Intent*.

A *Intent* representa a mensagem de uma aplicação enviada ao Sistema Operacional e leva consigo informações necessárias para que seja interceptada pelo aplicativo de destino. Em um aplicativo, sempre que é feito o uso é necessária a configuração do *Intent Filter* (Filtro de Intenções). É possível que além de invocá-las, a aplicação seja uma interceptadora de Intent's (Lecheta, 2010). Desta forma toda vez que uma nova Intent é iniciada, dependendo dos seus argumentos, poderá ser interceptada pelo aplicativo através de um *Intent Filter*.

[7] Encontrado em: http://developer.android.com

10.3.8 Principais arquivos e diretórios de um Projeto Android

Um projeto Android tem sua estrutura basicamente formada por quatro diretórios. A estrutura básica dos diretórios descritos abaixo segue o padrão do *plug-in* do Android disponível para o Eclipse (Lecheta, 2010):

- **src/:** nesta pasta estão contidos os códigos fontes em Java, neles são feitos modificações que envolvem a parte de lógica e utilização da API;
- **res/:** nela encontram-se três subpastas, *layout*, *drawable*, *values*, juntos são responsáveis por recursos como as imagens utilizadas no aplicativo, os arquivos de *layout* e arquivos de internacionalização.;
- **drawable/:** na pasta *drawable* encontramos arquivos de imagens que são compilados. Há possibilidade de três tipos de resolução: alta (*drawable*-hdpi), media (*drawable*-mdpi), baixa (*drawable*-ldpi);
- **gen/:** Este diretório tem a responsabilidade de fornecer uma comunicação entre as pastas src/ e res/;
- **values/:** a internacionalização do aplicativo através da organização dos textos da aplicação em um único arquivo (*strings*.xml), definição de cores (*color*.xml), recursos que representam estilos (*styles.xml*).
- Sabendo os principais diretórios de um projeto, destacam-se brevemente alguns dos mais importantes arquivos de um projeto Android (Felker; Dobbs, 2011):
- **AndroidManifest.xml :** mantém o controle de todas as necessidades da aplicação. Nele estão contidas informações como a *API Level* mínima suportada, o nome do aplicativo Java, nome da(s) *Activiti*(es) e seus respectivos *Intent-Filter*, além de todas as permissões necessárias para o pleno funcionamento da aplicação. Como exemplo de permissões pode-se citar: *permission*.INTERNET, *permission*.BLUETOOTH, *permission*.VIBRATE;
- **Main.xml :** nele são inseridas informações necessárias para a definição dos layout de interface gráfico com o usuário;
- **R.java :** tem o papel de fazer a comunicação entre as pastas /src e /res.
- **default.properties :** contém informações do projeto como por exemplo o destino de compilação, é utilizado em conjunto com o ADT (*Android Development Tools*) e o Eclipse.
- **strings.xml :** localizado no diretório /res/values, contém valores strings que serão utilizadas no projeto. Útil para a internacionalização do aplicativo.

Todos os diretórios e arquivos descritos acima são gerados automaticamente após a criação do projeto Android. Entender a funcionalidade dos principais arquivos de um projeto Android se torna importante para um resultado mais satisfatório na eficiência do aplicativo.

10.4.1. Elementos da interface de usuário

A maioria das aplicações em Android invariavelmente precisão comunicar diretamente com o usuário. Esta seção apresenta os elementos da interface de usuário disponíveis no SDK Android. Alguns elementos são especializados na entrada de dados, enquanto outros são especializados na saída de dados.

O SDK Android tem um pacote Java chamado *android.view*. Este pacote contem uma variedade de interfaces e classes voltadas para adicionar elementos a uma tela. A classe *View* é a base do módulo de criação de componentes da interface de usuário e de *layouts* numa aplicação Android.

Um elemento do tipo *layout* faz parte do pacote *andoid.widget*, é um objeto *View*, mas, ele não adiciona elementos na tela. Em vez disso, ele organiza outros controles, e determinar como estes controles-filhos deverão ser dispostos na tela.

10.4.2. TextView

Elemento que mostra na tela informações no formato texto, bem como possibilita a conexão do texto com navegadores, aplicações de e-mail, mapas etc., como ilustrado na Figura 10.7.

Figura 10.21 - Componente TextView
Fonte: (Condey; Darcey, 2001)

10.4.3. EditText

Controle que lida com a entrada de texto. Como apresentado na Figura 10.8, o controle *EditText* pode assumir várias formas.

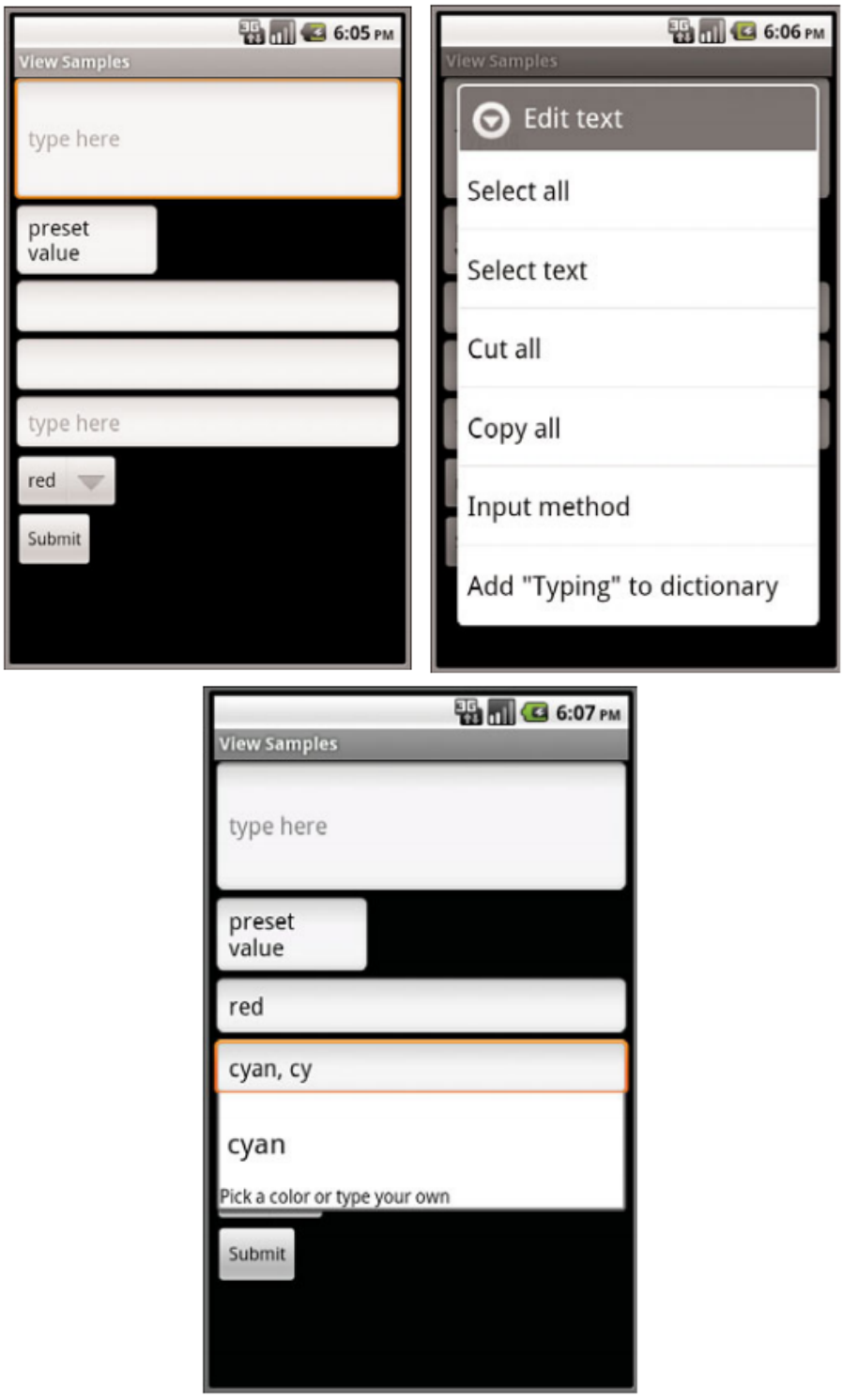


Figura 10.8 – Formas diferentes do componente *EditText*
Fonte: (Condey; Darcey, 2001)

10.4.4. Spinner

O controle *Spinner* possibilita que a entrada do usuário seja limitada por um número pré-definido de opções, como mostra a Figura 10.9.

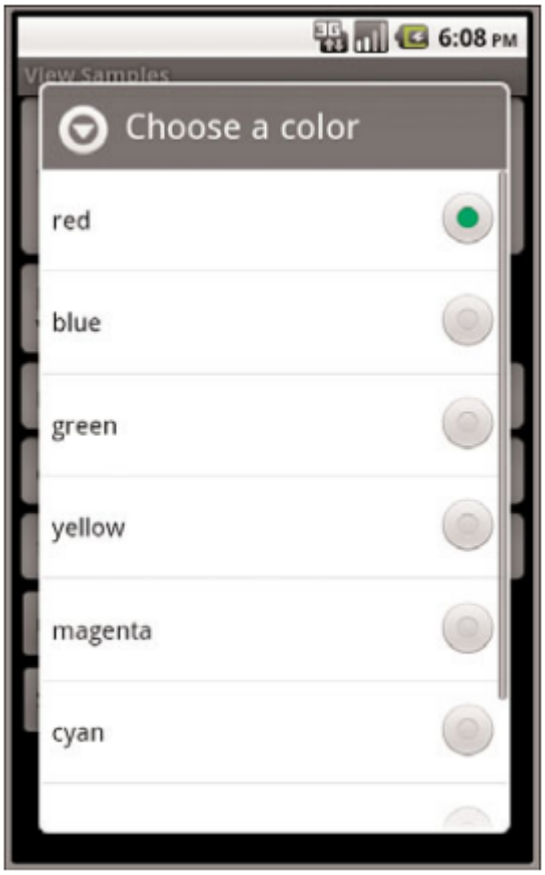


Figura 10.9 – Formas diferentes do componente *EditText*
Fonte: (Condey; Darcey, 2001)

10.4.5 Botões, *Check Box* e *Radio Groups*

Os botões estão geralmente associados a ações. Já os componentes *Check Box* estão associados a informações que pode ter dois estados. A variedade de formatos destes controles pode ser vista, a seguir, na Figura 10.10.

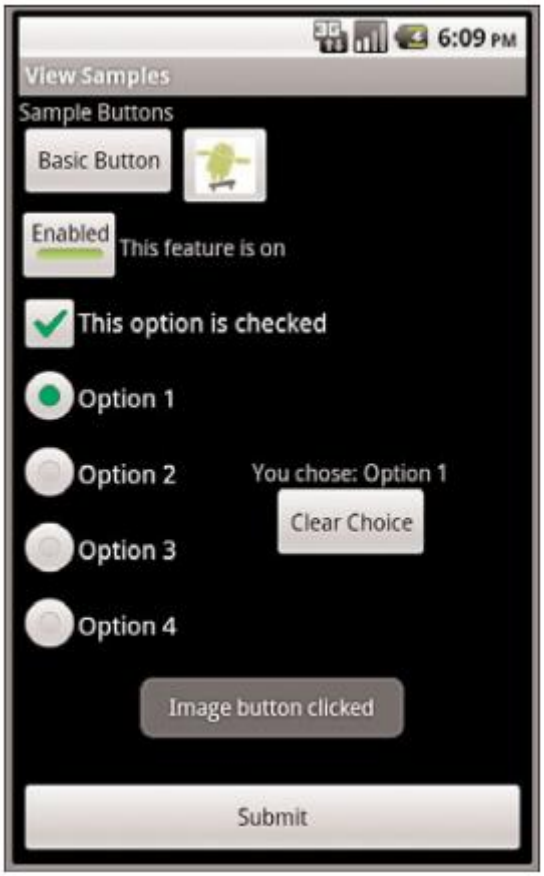


Figura 10.10 – Botões, Check Box, Radio Group
Fonte: (Condey; Darcey, 2001)

10.4.6 Obtendo datas

Para validar a inserção de datas, o Android disponibiliza os componentes apresentados na Figura 10.11.

Figura 10.11 – Controles para obtenção de datas
Fonte: (Condey; Darcey, 2001)

10.4.7. Controles indicadores

O Android disponibiliza uma série de controles que funcionam como indicadores de progressão de atividades (barras de progressão), horas, alarmes etc. A Figura 10.12 mostra alguns desses controles indicadores.

Figura 10.12 – Controles indicadores
Fonte: (Condey; Darcey, 2001)

10.4.8. Menus

Os menus são bastante utilizados para ajudar na navegação, para fornecer informação adicionar, para fornecer ajuda ou para mostrar opções disponíveis num dado momento. Como pode ser visto na Figura 10.13, o Android disponibiliza menus baseados em texto e menus baseados em ícones.

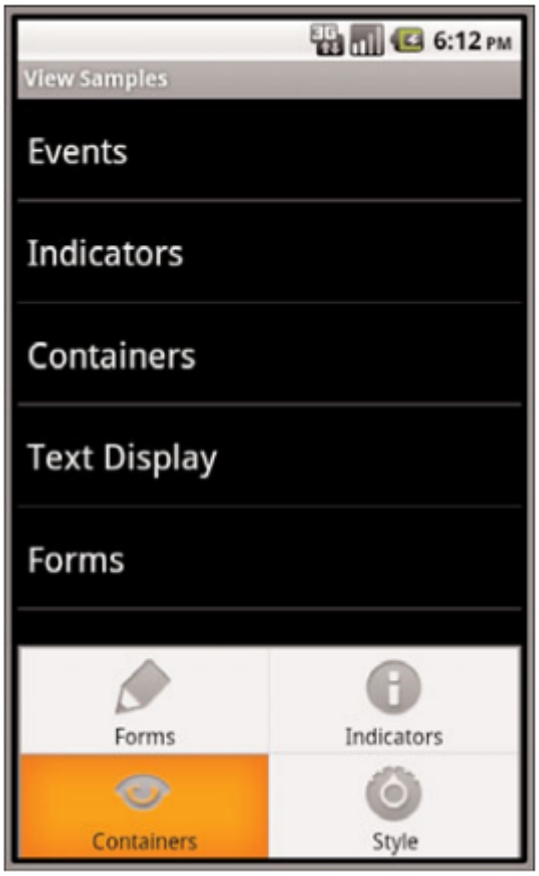


Figura 10.13 – Controles para obtenção de datas
Fonte: (Condey; Darcey, 2001)

10.5.1 SQLiteDatabase

Para a realização de diversas atividades disponíveis nos dispositivos móveis, surgiu a necessidade da criação de aplicativos com a capacidade de armazenar e manter o controle de grandes quantidades de dados. Como exemplo pode-se citar funções que necessitem armazenar informações para utilizá-las em outro momento (Rogers *et al*., 2009).

O Android fornece opções para a persistência de dados. Provedores de Conteúdos e Preferências Compartilhadas (*Shared Preferences*) são algumas das alternativas disponíveis para salvar informações, porém, para ocasiões em que a aplicação requer um mecanismo mais robusto e organizado para o armazenamento de dados, faz-se importante o uso do banco de dados (Merrifield, 2011). A escolha final irá depender das necessidades específicas. Atualmente, o Android oferece suporte completo ao SQLite (OHA, 2011).

O SQLite é um banco de dados poderoso e compacto, baseados em arquivos, ideais para dispositivos móveis. Quando integrado ao Android, o SQLite tem seus dados incluídos na pasta relativa ao projeto da aplicação. Cada aplicativo pode ter um ou mais banco de dados, as suas informações são privadas, ou seja, são acessadas apenas pelo aplicativo que as originou (Conder, 2010).

10.5.2 Mapas e Localização

Desde que os dispositivos móveis começaram a prover serviços de localização baseados em coordenadas geográficas, foi prevista uma nova era de aplicativos (Rogers *et al*., 2009).

A maneira como os aparelhos detectam sua localização é feita através das torres de telefonias próximas ou via conexões a internet Wi-Fi (Merrifield, 2011). Outra maneira de obter a localização é através do *Global Positioning System* (GPS). Sabe-se que existem 31 satélites que giram em torno do planeta, assim, o GPS que originalmente foi desenvolvido para fins militares, contribui para uma boa precisão na posição do aparelho (Burnnete, 2010).

É importante mencionar algumas limitações no uso de recursos de mapas e localização, entre os fatores responsáveis para uma possível falha ou erro citam-se: requisição do recurso em locais onde os sinais GPS não são captados, queda na conexão e consequentemente não atualização do mapa, além de uma menor precisão quando comparados a aparelhos GPS convencionais (Rogers *et al.*, 2009).

Na plataforma Android são oferecidos recursos poderosos para o desenvolvimento de aplicativos com funcionalidades de exibição de mapas e localização, tais recursos são acessados através da *Google Maps API*. Na próxima seção são abordadas as suas principais características desta API.

10.5.3. API do Google Maps para Android

A *Google Maps API* para a plataforma Android, atualmente na versão 4.0 API *Level* 15, oferece diversos meios para o desenvolvimento de aplicativos Android integrados com o Google Maps. Controle de zoom, modo de visualização (rua ou satélite), manuseio com coordenadas geográficas (latitude e longitude) são algumas funcionalidades que podem ser utilizadas com a API.

No contexto de aplicação utilizando a API *Google Maps*, destacam-se como principais componentes:

- **MapView:** componente gráfico fundamental responsável pela imagem do mapa;
- **GeoPoint:** classe responsável por conter as coordenadas (latitude e longitude);
- **MapController:** funciona como um controlador de funções do *MapView*;
- **Overlay:** tem como principal característica a realização de sobreposição de imagens no *MapView*;
- **GeoCoder:** classe capaz de transformar um endereço de rua ou descrição de um local em Coordenadascgeográficas, ou vice-versa;
- **LocationManager:** responsável por fornecer acesso aos serviços de localização do sistema.

O *MapView* é um componente essencial, pois torna possível a visualização do mapa de qualquer área do mundo através do *Google Maps*. Mais precisamente, o *MapView* é a

classe da API do *Google Maps* responsável por mostrar o mapa na tela, responder ao usuário eventos como toque ou controle de zoom e oferecer recursos como sobreposição de imagens (ROGERS *et al.*, 2009).

Os modos de exibição de mapas são definidos pelos métodos `setStreetView()`, `setSatellite()` e `setTraffic()`. É possível alternar o mapa entre os modos exibição das ruas, exibição das ruas e condições de tráfego, e exibição do tipo satélite, como ilustrado na Figura 10.14.

Figura 10.14 - Modos de visualização de mapas
Fonte: (Google, 2011)

Cada localidade em um mapa pode ser representada por um par latitude/longitude. Para representá-las em um código é preciso utilizar a classe *GeoPoint*, que armazena o par de informações para indicar o endereço no *MapView*.

Os recursos oferecidos pela classe *MapController* são importantes para a manipulação de algumas características do *MapView*, como, por exemplo, a centralização em um *GeoPoint* ou o controle de *zoom*.

Geralmente, em aplicações utilizando mapas o usuário sentirá a necessidade de criar marcadores para identificar localidades, ou seja, estas imagens ou marcadores irão sobrepor o mapa. Uma das melhores características da *Google Maps API* é a possibilidade de sobreposição de imagens sobre o *MapView*. A classe responsável pela funcionalidade em questão é a *Overlay* (Merrifield,2011).

Através dos recursos da classe *Overlay*, é possível construir aplicações que oferecem a função de criar marcadores, além da possibilidade de tratar eventos como o controle de toque delimitando os limites da imagem no *MapView*.

Utilizando a classe *Overlay*, a implementação necessária para identificar quando o usuário toca ou clica em um *Overlay* requer um trabalho árduo e com baixa precisão (Lecheta, 2010). Uma solução eficaz pode ser encontrada no uso da classe *ItemizedOverlay*. Dentre outros benefícios, esta classe, de maneira automática, faz todo o controle de clique nas imagens do aplicativo.

Em algumas situações, pode ser necessário ter conhecimento do local/endereço de onde o dispositivo se encontra. Esta necessidade é sanada pela *Google Maps API* através das classes *LocationProvider, Location*, e *LocationManager*. A classe *LocationManager* fornece serviços de localização capazes de obter atualizações da localização geográfica do aparelho através de intervalos regulares (GOOGLE, 2011).

Além das classes de localização, pode-se citar também a classe *GeoCoder*, que é capaz de realizar o processo de transformar uma coordenada, isto é latitude/longitude, em um endereço - Bairro, Rua, Avenida, CEP -, ou vice-versa. Vale ressaltar que a quantidade de detalhes do resultado obtido através do *GeoCoder* varia de cada endereço, obtendo assim, resultados com riqueza de detalhes distintas (GOOGLE, 2011).

10.6. Conclusão

Caso queira criar um diferencial no mercado promissor de desenvolvimento de aplicações para dispositivos móveis, os profissionais de Engenharia de Software necessitam compreender o contexto em que os usuários utilizam tais aplicações, suas necessidades e limitações.

Este capítulo apresenta os princípios que devem nortear o projeto de interface de usuários para aplicações que serão executadas em equipamentos móveis. Além disso, abordou-se a plataforma Android, uma das soluções com maior grau de aceitação no mercado e na academia para a criação destas aplicações. Foram discutidos os componentes de uma aplicação criada no Android, a arquitetura da plataforma, e os componentes de interface de usuário destinados à entrada e à saída de dados.

10.7. Referências Bibliográficas

B'FAR, R. Mobile Computing Principles: Designing and Developing Mobile Applications with UML and XML. Cambridge: CAMBRIDGE UNIVERSITY, 2004.

BURNETTE, E. Hello, Android: Introducing Google's Mobile Development Plataform. 3ª. ed. Dallas: LLC, 2010.

CONDEY, S; DARCEU, L. Android Wireless Application Development. Addison-Wesley. 2011.

GOOGLE. Android Developers. Android Developers, 2007. Disponivel em: <http://developer.android.com>. Acesso em: 21 novembro 2011.

LECHETA, R. R. Google Android: Aprenda a Criar Aplicações para Dispositivos Móveis com o Android SDK. 2ª. ed. São Paulo: Novatec, 2010.

MACK, R. S. Sistema de recomendação baseado na localização e perfil utilizando a plataforma Android. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2010.

NIELSEN, J. Usability engineering. University Cambridge. San Francisco. 1993.

NIELSEN, J. Defer Secondary Content When Writing for Mobile Users. 2011. Disponível em http://www.useit.com/alertbox/mobile-content.html.

OPEN HANDSET ALLIANCE. Open Handset Alliance. Open Handset Alliance, 2007. Disponivel em: <http://www.openhandsetalliance.com>. Acesso em: 19 novembro 2011.

ROGERS, R. et al. Android Application Development. Sebastopol: RepKover, 2009.

WEISER, M. (1991). "The computer for the 21st century." Scientific American 265: 66-75.

YAMABE, Tetsuo; TAKAHASHI, Kiyotaka; NAKAJIMA, Tatsuo. 2008. Design issues and an empirical study in mobility oriented service development. In Proceedings of the 1st workshop on Mobile middleware: embracing the personal communication device (MobMid '08). ACM, New York, NY, USA

Capítulo 11

Utilizando o Framework JQuery

Ivan Rodrigues, Micarlla P. Melo

Resumo

JQuery é uma das bibliotecas javascript mais utilizada em todo mundo para o desenvolvimento de aplicativos web. Seu uso permite que o desenvolvedor escreva menos linhas de código e ainda assim consiga fazer páginas mais interativas e agradáveis para o usuário. Este minicurso foi proposto com o intuito de familiarizar estudantes e profissionais com essa biblioteca que já tem um espaço consolidado no desenvolvimento de sites e sistemas.

11.1 Dados Gerais

11.1.1 Objetivo do curso

O curso Utilizando o Framework JQuery tem como objetivo geral familiarizar estudantes e profissionais com a biblioteca jQuery. Tendo como objetivo específico apresentar como fazer o download da biblioteca, adicionar essa biblioteca nos projetos, expor os principais comandos, exemplos e mostrar as vantagens de se utilizar esse framework por meio de comparativos entre códigos que se utilizam jQuery e que utilizam somente javascript.

11.1.2. Tipo do curso

O curso Utilizando o Framework JQuery terá uma abordagem teórica contendo exemplos que são necessários para tornar o conteúdo mais intuitivo e facilitar a compreensão dos tópicos que serão apresentados.

11.1.3. Tratamento dado ao tema

Inicialmente será apresentado o que é o jQuery, qual é a sua origem e como ele vem sendo utilizado nos principais sites do mundo. Em seguida, será feita uma breve revisão de HTML e da linguagem JavaScript, já que são conteúdos necessários para entender o que será abordado durante o curso. Posteriormente, serão mostrados os principais comandos do

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

jQuery e os respectivos exemplos que servirão de base para uma melhor compreensão de como esses comandos são utilizados.

11.1.4. Perfil desejado dos participantes

Profissionais e estudantes interessados em conhecer a biblioteca jQuery e conhecimento básico em HTML e JavaScript.

11.1.5. Infraestrutura física necessária para a apresentação

Equipamentos: microcomputador, projetor multimídia, quadro branco, pincéis, apagador.

11.2. Estrutura prevista detalhada do texto

O jQuery é uma biblioteca JavaScript de código fonte aberto criada com o objetivo de tornar simples e fácil a maneira de escrever aplicações web JavaScript, facilitando a manipulação e o acesso aos elementos desta linguagem. Além disso, o jQuery incrementa nas páginas web dinamismo e interatividade com os usuários.

A biblioteca jQuery foi criada em 2006 por Jhon Resig, um desenvolvedor americano e experiente em JavaScript e autor do livro *Pro JavaScript Techniques*. Em seu livro *jQuery in Action*, ele afirma: "O foco principal da biblioteca jQuery é a simplicidade". Esta simplicidade permite que não seja necessário um profundo conhecimento em JavaScript. (SILVA, 2008)

O uso do jQuery acrescenta um melhor design, usabilidade e acessibilidade em uma página web. Com o jQuery pode-se adicionar efeitos visuais e animações, aumentar a interatividade, simplificar tarefas específicas de JavaScript, buscar informações no servidor sem a necessidade de recarregar a página etc. (SILVA, 2008) O seu uso é mais provável em sites de negócios, compras ou sites relacionados à tecnologia. Alguns sites que se utilizam do jQuery são: http://www.duchyoriginals.com/, http://www.sacredjune.com/#bio, http://grooveshark.com/ http://www.samsung.com/br/#tv-audio-video-home, http://www.worldofmerix.com/.

Atualmente, jQuery é a biblioteca JavaScript mais utilizada em páginas web. De acordo com a BuiltWith.com, que é uma empresa australiana que acompanha as tendências tecnológicas e os conhecimentos web, dos 10.000 melhores sites da Internet, cerca de 50% destes utilizam jQuery em sua implementação. Isso sugere que o jQuery contribui favoravelmente na qualidade do site. Na figura 11.1, observamos um gráfico que representa o crescimento do uso do jQuery nos 10.000 principais sites no período de 2008 até 2011.

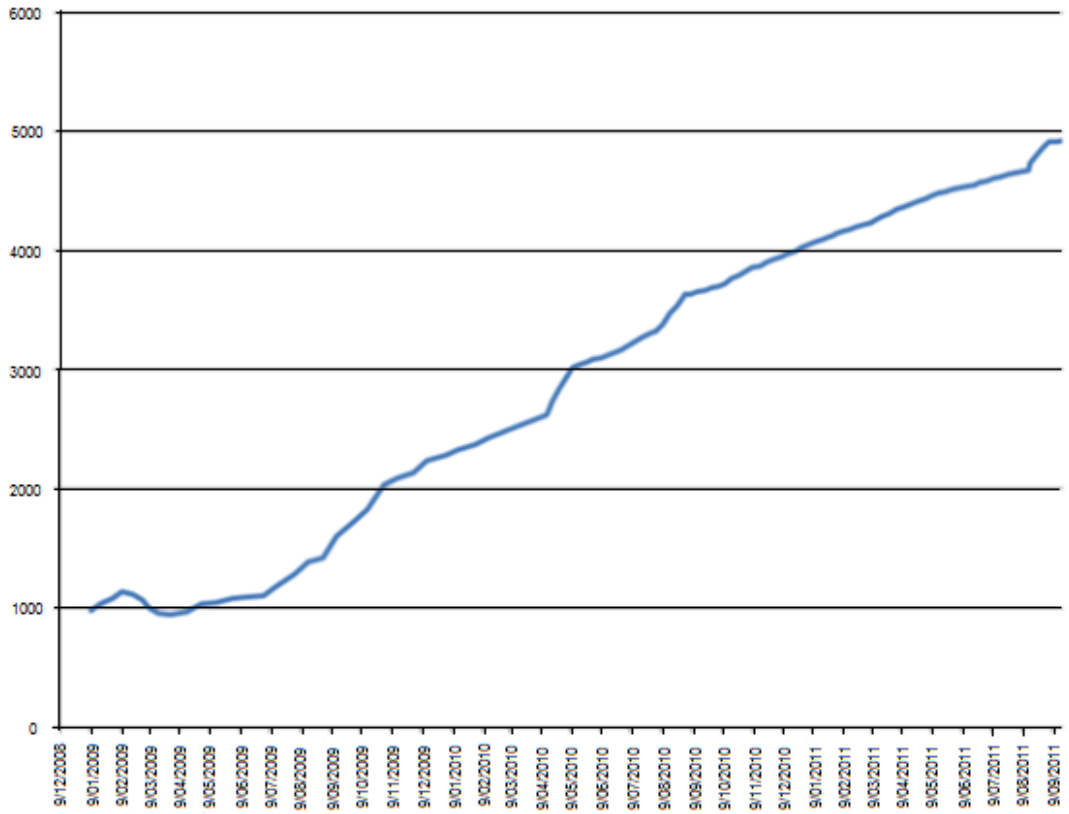


Figura 11.1. Representação gráfica do uso do jQuery em 10.000 sites desde 2008.

Os plug-ins jQuery proporcionam muitas funcionalidades úteis para uma página web, tornando-a mais interativa e dinâmica com o usuário. Entre as diversas funcionalidades disponíveis, as mais utilizadas pelos programadores são: jQuery User Interface (Interface do Usuário) que proporciona abstrações de interação e animação, efeitos avançados e de alto nível e widgets personalizável (jqueryui.com, 2010); jQuery Form, que permite atualizar formulários HTML para usar AJAX; e, jQuery Cycle, que suporta diferentes tipos de efeitos de transição. Além desses, existem funcionalidades para validação, compatibilidade com navegadores mais antigos e muitas outras funções.

Na figura 11.2, é apresentado um gráfico contendo as porcentagens das funcionalidades mais frequentemente encontradas no top milhões de sites principais pesquisados pela BuiltWith.com. Nele podemos observar que os plug-ins mais utilizados são o jQuery User Interface (UI), jQuery Form e o jQuery Cycle.

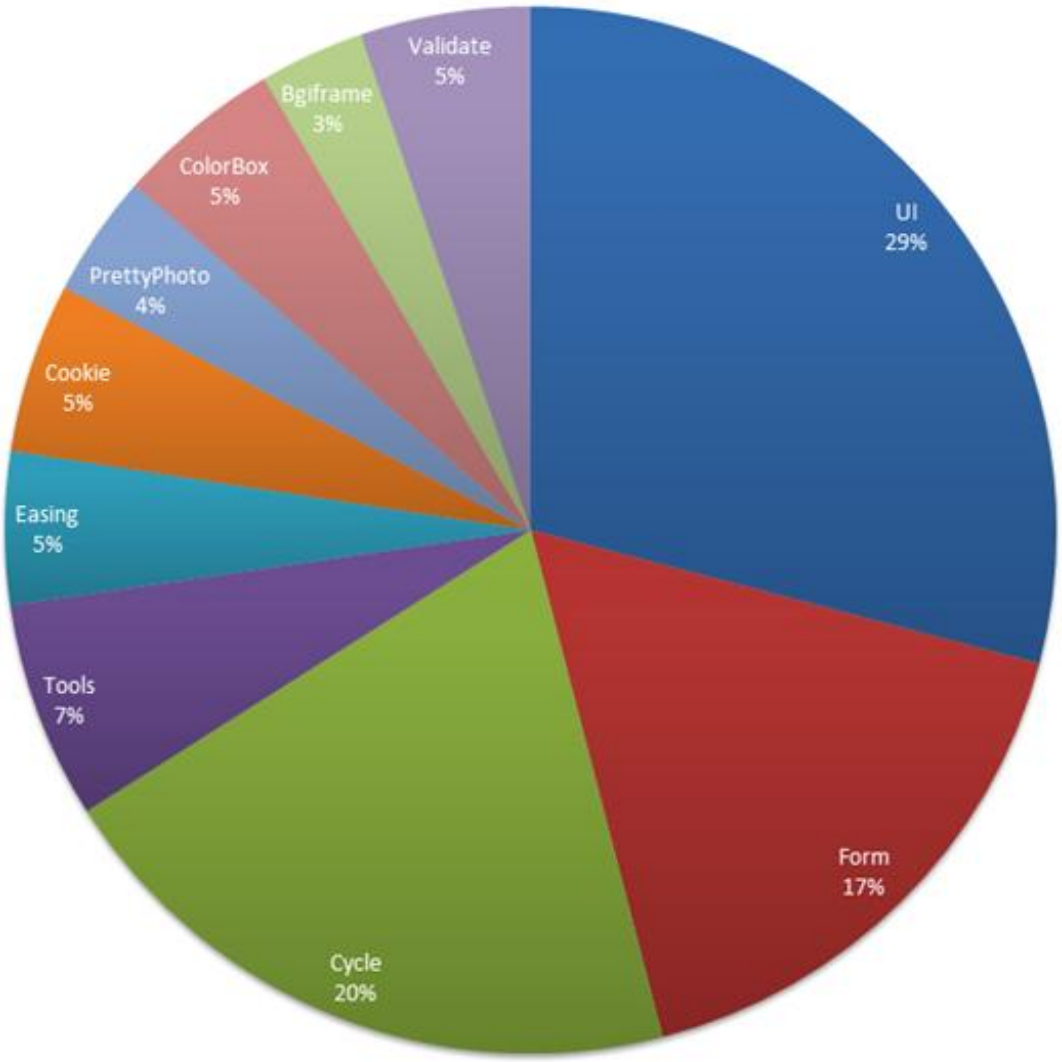


Figura 11.2. Os 10 plug-ins jQuery mais utilizados no top milhões de sites principais.

A seguir, o conteúdo programático oferecido pelo curso:

11.3. Introdução

11.3.1. HTML

11.3.1. JavaScript

11.4. Adicionando a biblioteca

11.4.1. Fazendo o download da Biblioteca

11.4.2. Adicionando a biblioteca ao projeto

11.4.3. Inserindo biblioteca no código

11.5. Criando primeiro Exemplo

11.6. Jquery e padrões web

11.7. Entendendo a estrutura do código

11.3. Introdução

Antes de começarmos a falar do jquery iremos fazer uma breve revisão de HTML e javascript para esclarecer alguns conceitos que serão utilizados pela biblioteca. O jquery é uma biblioteca escrita em javascript que é utilizada para manipular elementos HTML, por isso é necessário que o estudante tenha o mínimo de familiaridade com essas duas tecnologias.

11.3.1. HTML

Hyper **T**ext **M**arkup **L**anguage, mais conhecido como HTML, é uma linguagem de marcação comumente utilizada para implementação de sites e sistemas web. No HTML o programador determina quais elementos estarão presentes num site como tabelas, vídeos, textos e tudo que até hoje você viu em um navegador. O código HTML é composto por

tags que descrevem a forma e comportamento básicos de cada elemento. Tags são marcações que podem ser simples ou duplas e podem possuir atributos que modificam seu comportamento padrão. A seguir alguns exemplos de tags (Figura 11.3):

```
<div id='quadrado'></div>(exemplo de tag dupla com atributo 'id' de valor 'quadrado')
<BR/>(exemplo de tag simples)
```

Figura 11.3. Exemplos de tags HTML.

A estrutura de um documento HTML é a seguinte, como apresentada na figura 4:

```
<HTML>
<HEAD>
<!-- Esse é um comentário HTML -->
<!-- Na tag HEAD colocamos tudo que queremos que carregue antes da página abrir -->
</HEAD>
<BODY>
<!-- Na tag BODY colocamos tudo que queremos que carregue no corpo da página  -->

</BODY>
</HTML>
```

Figura 4. Estrutura de um documento HTML.

11.3.2. JavaScript

JavaScript é uma linguagem de script disponível na maioria dos navegadores de internet que serve para tornar a página mais interativa. O HTML não é uma linguagem de programação, sendo assim não possui variáveis, comandos de repetição e comandos de decisão. O javascript foi criado para suprir essa limitação, permitindo tornar as páginas web mais interativas. Todo código javascript deve se localizar dentro da tag “<SCRIPT>” como no exemplo da figura 11.5 abaixo:

```
<HTML>
<HEAD>
</HEAD>
<BODY>

<SCRIPT>
var frase ='hello Javascript';
alert(frase+' , seja bem vindo!');

</SCRIPT>
</BODY>
</HTML>
```

Figura 11.5. Exemplo de código javascript.

Repare que o javascript é uma linguagem baseada no C, sendo preciso acabar todos os seus comandos com “;”. Repare também que a declaração da variável começa com a palavra reservada “var”. O javascript não precisa de declaração de tipo para as variáveis, mas os tipos de dados existem, tornando impossível a soma da palavra “5” com o valor 3 sem que antes haja uma conversão da string para inteiro. Por último, repare que a função “alert” (responsável por dar um alerta na página) possui um sinal de soma (“+”) no parâmetro. Esse comando é o responsável por concatenar strings no javascript.

Essas observações foram feitas porque as mesmas serão muito úteis quando estivermos trabalhando com o jquery. Sem essas informações o programador que não tem familiaridade com essas tecnologias não teria muita facilidade para acompanhar o curso.

11.4. Adicionando a biblioteca

11.4.1. Fazendo o download da Biblioteca

Para fazer o download a biblioteca vá ao site http://jquery.com/. Esse site além de ter a biblioteca para download é também uma fonte de informação completa sobre a biblioteca, tendo tutoriais, fóruns de dúvidas e a documentação completa. Na página inicial existem duas opções de download. Selecione a opção “PRODUCTION” e clique no botão de download (Figura 11.6). A versão “DEVELOPMENT” é uma versão descompactada para o programador que deseja fazer modificações no conteúdo da biblioteca. Depois que o download estiver completo salve o arquivo na pasta na qual serão feito os testes.

Figura 11.6. Página de download da biblioteca jQuery.

11.4.2. Adicionando a biblioteca ao projeto

Uma boa prática de programação web é separar os arquivos javascript em uma pasta exclusiva comumente chamada de “js”. Todos os exemplos desse minicurso referenciarão a biblioteca contida nesta pasta. Caso o estudante queira acompanhar os exemplos é recomendado que o mesmo crie uma pasta de exemplos e dentro desta crie uma pasta chamada “js”. Depois de criar as pastas copie a biblioteca baixada para dentro da pasta “js”.

11.4.3. Inserindo biblioteca no código

Para inserir a biblioteca na pasta é utilizado o comando de importação do javascript como no exemplo mostrado na figura 11.7 abaixo:

```
1  <HTML>
2  <HEAD>
3
4  <SCRIPT type="text/javascript" src="js/jquery.js"></SCRIPT>
5
6  </HEAD>
7  <BODY>
8  </BODY>
9  </HTML>
10
```

Figura 11.7. Exemplo para inserir a biblioteca jquery no código.

Note que a inclusão da biblioteca foi feita na tag HEAD. Isso ocorreu porque é necessário que a biblioteca esteja disponível antes que a mesma seja usada. Comandos jquery que sejam escritos antes da importação da biblioteca acarretaram em erros na página.

11.5. Criando primeiro Exemplo

Como primeiro exemplo criaremos um “alert()” simples que aparentemente não se diferem em nada do primeiro exemplo que criamos quando estávamos revisando javascript. Observe o exemplo da figura 11.8 abaixo:

```
1  <HTML>
2  <HEAD>
3  <SCRIPT type="text/javascript" src="js/jquery.js"></SCRIPT>
4
5  </HEAD>
6  <BODY>
7
8  <SCRIPT>
9  $(function(){
10
11 var frase ='hello Javascript';
12 alert(frase+' , seja bem vindo!');
13
14 });
15
16 </SCRIPT>
17 </BODY>
18 </HTML>
19
```

Figura 11.8. Exemplo criando um alert().

Ao abrir essa página, o resultado visualmente será o mesmo da página anterior. O que mudou foi a ordem com que os eventos ocorreram, mesmo que isso seja imperceptível.

O uso do comando "$(function(){});" é a forma que jquery usa para dizer ao navegador que execute isto quando a página estiver completamente carregada. Se dividirmos o comando anterior em partes o entendimento seria que "$( );" é uma função na qual o parâmetro é uma função sem nome que será executada quando a página for completamente carregada. Dessa forma, quando os comandos contidos nela forem chamados todos os elementos da página já estarão disponíveis.

11.6. Jquery e padrões web

Uma das vantagens de se usar jquery é a possibilidade de fazer uma separação de códigos estruturais de códigos de evento. Essa separação é considerada uma boa prática da programação web, pois seu uso permite que o desenvolvedor tenha uma visão mais clara do código o que acaba facilitando a detecção de erros e implantação de melhorias. A seguir veja um exemplo no qual essa separação não foi levada em conta (Figura 11.9):

```
1  <HTML>
2  <HEAD>
3
4  <SCRIPT type="text/javascript" >
5  function mostraMensagem(mensagem){
6  alert(mensagem);
7  }
8  </SCRIPT>
9
10 </HEAD>
11 <BODY>
12 <button onClick="mostraMensagem(Olá mundo);">Click aqui</button>
13 </BODY>
14 </HTML>
15
```

Figura 11.9. Exemplo sem a separação de códigos estruturais e de códigos de evento.

Note que nesse exemplo o evento "onClick" (evento chamado ao se clicar no elemento) se encontra no meio do código HTML e essa junção que se deve evitar. A seguir, na figura 11.10, será mostrado um código utilizando jquery no qual a separação conceitual é implementada:

```
<HTML>
<HEAD>
<SCRIPT type="text/javascript" src="js/jquery.js"></SCRIPT>

<SCRIPT>
    $(function(){

        $('button').click(
            function(){
                alert('Olá jquery');
            }
        );

    });

</SCRIPT>

</HEAD>
<BODY>
<button >Click aqui</button>

</BODY>
</HTML>

```

Figura 11.10. Exemplo de código se utilizando da separação de códigos estruturais e de códigos de evento.

Note que no código estrutural do HTML não existe nenhuma linha de javascript. O botão é declarado no corpo do site e a ação atribuída ao clique é descriminada no bloco javascript. Note que a ação atribuída ao evento do botão é declarada antes mesmo do botão existir, mas como foi visto anteriormente, o uso do comando jquery "$( )" permite escrever comandos que só serão executados quando a página estiver completamente carregada. Essa prática deixa o código mais legível e consequentemente mais fácil de encontrar erros.

11.7. Entendendo a estrutura do código

A estrutura dos comandos jquery talvez seja a maior causa de desistência de aprender a usar a biblioteca por parte dos desenvolvedores que não estão familiarizados. O código realmente não se parece com a maioria das linguagens e isso causa certa resistência no desenvolvedor iniciante nessa tecnologia. Para quebrar essa barreira iremos agora explicar essa estrutura, identificando cada item e sua utilidade. A seguir é apresentado um código que servirá de base para a explicação (Figura 11.11):

```
<script>
$(    //início do $
//como parâmetro é passada uma function

function(){//início da função que será chamada ao completar o carregamento
//conteúdo que será executado

//identificação do elemento o qual se pretende atribuir uma característica ou evento
//no exemplo é atribuído o evento de click
$('button').click(
//início do evento de click

//é passado para o evento de click uma função com as ações que o evento causará
function(){
alert('olá mundo');
}


);

}//fim da função chamada ao completar o carregamento

);//fim do $

</script>
```

Figura 11.11. Estrutura de um código jquery.

Grande parte dos códigos jquery iniciam com o "$( );" que, como já foi citado nesse curso, é um comando que diz o que irá acontecer quando o carregamento da página acabar similar à variável "window.onload" do javascript. Esse comando recebe como valor uma função que será executada quando a página for completamente carregada. Uma das características do jquery que causa muito estranhamento no programador iniciante é o fato de que em muitos casos funções são passadas como parâmetros. No exemplo isso acontece duas vezes, na função que é chamada ao acabar o carregamento e no evento de click do elemento button. Esse é um recurso existente no javascript e em outras linguagens, mas que teve seu potencial observado na construção do jquery.

11.7.1. Uso do cifrão

O '$' é utilizado tanto na função que executa quando o carregamento acaba quanto para identificar os elementos na página. A seleção de elementos acontece da seguinte maneira:

$('seletor do elemento').atributo_ou_evento();

A seguir, alguns exemplos para esclarecer o uso desse comando.

- Deixando um elemento "div" invisível:

  $('div').hide();

- Tornando visível um elemento "p" que atualmente é invisível:

  $('p').show();

- Adicionando um evento click a uma div de classe “quadrado”:

```
$(‘div    .quadrado’).click(function(){    $(this).css(‘background’,’#334455’);
});
```

- Chamando o evento de click do elemento anterior:

```
$(‘div .quadrado’).click();
```

- Salvando o atributo “value” de um elemento input de id “salario”:

```
var x = $(‘input #salario’).attr(‘value’);
```

- Inserindo valor no atributo value de um elemento input:

```
$(‘input’).attr(‘value’,’45’);
```

11.7.2. Adicionando funções a eventos

Para adicionar funcionalidades a eventos selecionamos o elemento o qual desejamos atribuir a funcionalidade e depois dizemos a qual evento desejamos atribuir a funcionalidade e por último passamos a função que irá executar quando o evento ocorrer. Como no exemplo a seguir:

```
$(‘#botao’).click( function(){ alert(‘teste’); });
```

O jquery tem implementação para todos os eventos que já são nativos do javascript (Por exemplo: click, hover, blur, change) e ainda implementa eventos que não são implementados nativamente. Alguns eventos permitem que atributos sejam passados, como por exemplo, o click que permite que um atributo com informações sobre o evento seja passado.

11.8. Seletores

Seletores são o meio pelo qual é possível acessar propriedades e eventos de elementos. O uso de seletores é um dos grandes ganhos do jquery em relação ao javascript puro, onde para acessar um elemento qualquer muitas vezes era preciso escrever comandos extremamente extensos para navegar entre os elementos. O uso de seletores torna o código jquery parecido com uma folha de estilo, o que facilita sua visualização. Muitas vezes haverá mais de um seletor que chegue ao mesmo elemento, no entanto, saber escolher bons seletores é uma prática que pode aumentar consideravelmente a velocidade de execução de uma página. Nos tópicos seguintes iremos falar dos tipos de seletores.

11.8.1. Seletores de classe

O atributo de classe foi criado no HTML inicialmente para definir na folha de estilo características comuns a elementos que pertencessem a esta, no intuito de reduzir código e aumentar o reuso. Da mesma forma, em jquery, os seletores de classe devem ser usados para atribuir características e funcionalidades comuns a todos os elementos daquela classe. O seletor de classe funciona da seguinte forma:

```
$(“.nome da classe”).função();
```

Repare que para selecionarmos uma classe utilizamos o ponto antes do nome da classe. Isso informa ao jquery que se trata de uma classe. Observe o exemplo abaixo, na figura 11.12, onde queremos que ao clicar em qualquer elemento da classe “box” queremos que sua cor mude para vermelho:

```
<html>
<head>
<script type="text/javascript" src="js/jquery-1.7.1.min.js"></script>
<script>
$(function() {
    $('.box').click(
    function(){
        $('.box').css('background','red');

    }
    );
});
</script>
<style>

.box {
background:#FFFF00;
width:200px;
height:200px;
}
</style>
</head>

<body>
<div class='box'></div>
<div class='box'></div>
<div class='box'></div>
</body>
</html>

```

Figura 11.12. Exemplo de seletor de classe para mudar a cor de um elemento da classe box.

11.8.2. Seletores de id

Id é um atributo disponível em todas as tags HTML. Sua única restrição de uso é que não haja mais de um elemento com o mesmo id, pois esse funciona como um código único e caso haja mais de um elemento com o mesmo id, a execução da página pode ser comprometida.

Para selecionar elementos a partir do id utilizamos o seguinte comando:

$(“#identificador”).função();

Note que ao invés do ponto que utilizamos na seleção de classe, utilizamos agora um “#”. A utilização de id é muito útil quando estamos tratando elementos que vem de um banco de dados, por exemplo, que geralmente possuem chave primária que são obrigatoriamente únicas.

11.8.3. Seletores de tags

Além de podermos selecionar elementos por classe e por id, podemos selecioná-los pela tag que os compõem. Por exemplo, se quiséssemos fazer com que ao clicar em qualquer tag de parágrafo “ ” o mesmo mudasse de cor poderíamos usar o seguinte comando como apresentado na figura 11.13:

```
<script>

$(function() {
    $('p').click(
        function() {
            $(this).css('background', 'yellow');
        });
});

</script>

```

Figura 11.13. Comando para mudar a cor quando clicar na tag .

A seleção de tags é similar a seleção de classes, no entanto deve ser usada com muito cuidado, pois ela será usada em comandos nativos do HTML, o que pode gerar mudanças em elementos indesejados.

11.8.4. Usando o seletor $(this)

O seletor "$(this)" é utilizado para, na chamada de um evento relacionado a um determinado elemento, referenciar o próprio elemento que originou o evento. Esse recurso é muito útil e serve tanto para modificar o elemento que sofreu o evento quanto para pegar informações sobre este. Imagine a situação na qual ao clicar em um elemento que este saia da tela. Pra fazer este tipo de operação utilizamos o $(this) como no exemplo abaixo (Figura 11.14):

```
<HTML>
<HEAD>
<SCRIPT type="text/javascript" src="js/jquery.js"></SCRIPT>

<SCRIPT>
    $(function(){

        $('button').click(
            function(){
                $(this).fadeOut();
            }
        );

    });

</SCRIPT>

</HEAD>
<BODY>
<button >Click aqui</button>

</BODY>
</HTML>

```

Figura 11.14. Exemplo para fazer com que um elemento saia da tela ao ser clicado.

11.8.5. Usando mais de um seletor

Para especificar melhor qual elemento estamos manipulando podemos utilizar mais de um seletor para chegar nesse. O exemplo da figura 11.15 mostra um caso em que queremos mudar a cor de todos os parágrafos de classe “texto” contidos numa div:

$(‘div p .texto’).css(‘background’,’#234345’);

Além de podermos utilizar os seletores citados anteriormente, podemos usar seletores nativos do jquery como os exemplos a seguir:

```
<html>
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript"  src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
<script type="text/javascript">
$(function(){
$('li').css('color','red');
$('li:first').css('color','green');
$('li:last').css('color','blue');
$('li:nth-child(3)').css('color','yellow');
$('li:eq(1)').css('color','gray');
$('li a[title=titulo]').css('color','black');
});
</script>
<style type="text/css" media="all">

</style>

</head>
<body>
<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
<li><a href="#" title="titulo">Item 5</a></li>
<li>Item 6</li>
</ul>
</body>
</html>

```

Figura 11.15. Exemplo para mudar a cor dos parágrafos de classe “texto”.

11.8.6. Mudando valor de atributos

Atributos de um elemento HTML são valores que modificam suas propriedades padrão. Manipular valores de atributos é uma atividade corriqueira no cotidiano de um programador web. Quando queremos, por exemplo, atribuir um valor a um campo de um formulário depois de uma operação, mudamos o campo “value” deste. Para facilitar a manipulação dos atributos o jquery tem o método “.attr”. Observe o exemplo para entender seu funcionamento:

$(“input #valor”).attr(“value”,”50”);

Note que o primeiro passo é selecionar o elemento que queremos mudar o atributo. Os parâmetros do método “.attr” são o nome do atributo e o valor que desejamos atribuir a ele.

Além da operação de atribuição, é possível também pegar o valor atual dos atributos de um elemento. Observe o exemplo abaixo:

var valor = $(“input #valor”).attr(“value”);

Nesse exemplo o valor do atributo “value” é salvo na variável.

11.8.7. Manipulando o estilo dos atributos

Similar à manipulação de atributos, o jquery possui um método para manipulação do *css* dos elementos. Assim como o comando de manipulação de atributos, o método “.css” recebe 2 argumentos, sendo o primeiro a propriedade que se deseja mudar e o segundo o novo valor, como no exemplo a seguir:

$(“ #box”).css(“background”,”#FF2200”);

No exemplo anterior estamos mudando a cor de fundo de um elemento que possui id com valor igual a “box”.

Como no método de mudança de atributos, o método de mudança de css também permite que peguemos o valor atual de uma determinada propriedade. Para isso é preciso passar apenas o nome da propriedade que se deseja pegar o valor como no exemplo a seguir:

var valor = $(“#box”).css(“background”);

Esse recurso é usado com muita frequência pela sua praticidade em relação à forma como esse tipo de operação é feita sem o uso de jquery. Um exemplo desse uso é quando estamos dimensionando elementos dinamicamente em uma página e precisamos saber qual é o tamanho do container no qual iremos inseri-los.

11.9. Eventos

Ao longo do curso vimos algumas chamadas e atribuições de eventos relacionados a elementos. Todos os eventos, no jquery na verdade, são chamadas do comando bind. Dessa forma, a maneira nativa de chamar o evento de click seria como mostra a figura 11.16:

```
1  <script>
2
3  $(function(){
4
5          $('button').bind('click',
6              function(){
7                  $(this).fadeOut();
8              }
9          );
10
11     });
12 </script>
13
```

Figura 11.16. Exemplo de como chamar o evento click.

No entanto a maioria dos eventos possui atalhos, como por exemplo, o ".click()" que vimos em exemplos anteriores. Os próximos tópicos serão dedicados a explicar e demonstrar o uso de alguns desses eventos.

Para facilitar a compreensão dos eventos iremos utilizar um recurso do javascript que torna a leitura do código jquery mais legível: o armazenamento de funções em variáveis. O javascript permite que variáveis armazenem funções completas. Esse recurso é um dos que tornou possível a existência do jquery. Para entender como isso funciona observe o código na figura 11.17:

```
1  <script>
2  var comando = function(){
3
4  };
5
6  $(function(){
7
8          $('button').click(comando);
9
10     });
11
12 </script>
13
```

Figura 11.17. Exemplo do armazenamento de funções em variáveis.

11.9.1. Click

O evento de click é um dos mais comumente usados em aplicações web. Ele é acionado quando o elemento que contem o evento implementado é clicado. O evento click possui como atributo uma função que será executada assim que o evento ocorrer. Observe o exemplo na figura 11.18:

```
<html>
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript"  src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
<script type="text/javascript">
$(function(){
var count = $('#nav li').size()+1;

$('a#add').click(function(){

$('<li>Item '+count+'</li>').appendTo('#nav');

count++;
});

$('a#remove').click(function(){
if(count>1){
$('li:last').remove();

count--;
}
});


});
</script>
```

```

<style type="text/css" media="all">

</style>

</head>
<body>
<ul id="nav">
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
</ul>
<a href="#" id="add">Adicionar</a>
<a href="#" id="remove">Remover</a>

</body>
</html>

```

Figura 11.18. Exemplo de um código usando o evento click.

11.9.2. Hover

O evento hover é chamado quando o cursor do mouse é posto sobre o elemento que possui esse evento implementado. O evento hover possui como atributos duas funções anônimas, a primeira referente quando o cursor do mouse é posto sobre o elemento e a segunda referente à quando o cursor do mouse é retirado de cima do elemento.

A função que é acionada ao se retirar o cursor do mouse do elemento não tem implementação obrigatória, no entanto, caso o cursor passe sobre o elemento, a ação que ocorreu será permanente, até que outro evento a modifique. A seguir, na figura 11.19, é mostrado um exemplo de uso do evento hover.

```html
<html>
<head>
 <meta charset="utf-8" />
 <title></title>
 <script type="text/javascript"  src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
<script type="text/javascript">
var botaCursor = function(){
 $(this).css("background","green");

}

var tiraCursor = function(){
 $(this).css("background","yellow");

}

$(function(){
        $('#box').hover(botaCursor,tiraCursor);

});
</script>
```

```html
<style type="text/css" media="all">
 #box{
 background:red;
 width:300px;
 height:300px;
 position:relative;
 }

</style>

</head>
<body>
 <div id="box"></div>

</body>
</html>
```

Figura 11.19. Exemplo de como utilizar o evento hover.

11.9.3. Toggle

O evento toggle é ativado ao clicar em um elemento, mas se difere do click porque seu conceito pode ser entendido, traduzindo para o português como marcar. Dessa forma o evento toggle possui dois estados: marcado e desmarcado. Os atributos desse método jquery são duas funções que teriam o papel de marcar e desmarcar determinado elemento.
A seguir visualize um exemplo de utilização do toogle (Figura 11.20):

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
<script>
    $(function(){
    $('#box').toggle(
    function(){
    $(this).css('background','#00ffff');
    },
    function(){
    $(this).css('background','#ffff00');
    }

    );
    });
</script>

<style>
#box{
background:#FFFF00;
width:200px;
height:200px;
}

</style>
</head>

<body>
<div id='box'></div>
</body>
</html>

```

Figura 11.20. Exemplo do evento toggle.

Nesse exemplo atribuímos atributos CSS à nossa div e ao clicar uma vez (marcar) a primeira função foi chamada, quando clicamos pela segunda vez (desmarcar) a segunda função foi chamada. Dessa forma o jquery cria um efeito de marcação nos elementos.

Note que nesse exemplo não chamamos o jquery da pasta local. Ao invés disso, pegamos a biblioteca online no servidor do Google. Essa é uma prática muito comum em aplicações on-line, mas que precisa ser feita com prudência em relação à versão que está sendo utilizada na sua aplicação.

11.9.3.1. ToggleClass

Existem alguns atalhos em relação ao toggle e um dos mais usados é o toggleClass. Esse método marca e desmarca algum elemento, mas ao invés de detalharmos como será a marcação, ele apenas alterna entre adicionar e remover uma classe css. Para melhor visualizar o uso, observe o exemplo na figura 11.21:

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
<script>
$(function(){

$('#box').click(function(){

$(this).toggleClass('marcado');

});

});
</script>

<style>
#box{
background:#FFFF00;
width:200px;
height:200px;
}

.marcado{
border:3px solid;
}
</style>

</head>
<body>
<div id='box' ></div>
</body>
</html>

```

Figura 11.21. Exemplo do uso do método toggleClass.

11.9.3.2. slideToggle

O método slideToggle é mais um tipo de marcação utilizado pelo jquery. Diferente das formas que foram apresentadas anteriormente, esse método marca o elemento alvo fazendo o mesmo aparecer e desaparecer, por esse motivo é necessário utilizar outro elemento para disparar a marcação, pois caso o elemento alvo suma não será possível clicar nele, por exemplo. Visualize o exemplo abaixo (Figura 11.22):

```
<html>
<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
<script>
$(function(){

$('#gatilho').click(function(){

$('#box').slideToggle();

});

});
</script>

<style>
#box{
background:#FFFF00;
width:200px;
height:200px;
}
#gatilho{
background:#FFFF00;
width:200px;
height:20px;
}
.marcado{
border:2px solid;
}

</style>

</head>
<body>
<div id='gatilho'>click aqui</div><br/>
<div id='box' ></div>
</body>
</html>

```

Figura 11.22. Exemplo de código com o método slideToggle.

Nesse exemplo, ao clicarmos na div que tem o atributo id com valor “gatilho” a div de id igual a Box aparece e desaparece deslizando na tela.

11.9.3.3. Live

Quando vinculamos eventos aos elementos, esses eventos só funcionam para elementos que estavam inicialmente na página, ou seja, elementos que sejam criados em tempo de execução não possuem as funcionalidades atribuídas a eles. Para solucionar esse problema usamos o método live, que não está relacionado a nenhum evento específico, no entanto pode adicionar eventos a elementos criados em tempo de execução. No exemplo da figura 11.23, mostramos um caso onde adicionamos novos elementos a uma div e queremos que ao clicar em um deles, o elemento criado deixe de existir.

```
<html>
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript"  src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
<script type="text/javascript">
$(function(){
var count = $('#nav li').size()+1;

$('a#add').click(function(){

$('<li>Item '+count+'</li>').appendTo('#nav');

count++;
});

$('li').live('click',function(){
$(this).remove();
})



});
</script>
<style type="text/css" media="all">
li{cursor:pointer;}
</style>

</head>
<body>
<ul id="nav">
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
</ul>
<a href="#" id="add">Adicionar</a>

</body>
</html>

```

Figura 11.23. Código mostrando o uso do método live.

11.10. Métodos de Animação

Um dos maiores atrativos do jquery são seus métodos de animação. A popularização de banners e outros plug-ins animados feitos com jquery aconteceu rapidamente e hoje em dia, a maioria dos bons sites possui alguma animação feita com essa biblioteca. A grande vantagem de fazer uso desses recursos de animação é que eles não possuem tecnologias adicionais e fechadas como as animações em Flash e por isso o desempenho da página é aumentado. Além disso, códigos feitos em jquery são visíveis aos visitantes, tornando o compartilhamento de código e aprendizado de novas ideias mais acessíveis.

Como já foi dito antes, o jquery é uma biblioteca escrita em javascript. Dessa forma, todos os métodos de animação que a biblioteca possui poderiam ser feitos com javascript, no entanto, esses efeitos não tem uma implementação trivial e o jquery torna a utilização desses efeitos algo extremamente simples.

Nos próximos tópicos iremos abordar alguns dos métodos de animação disponíveis na biblioteca.

11.10.1. FadeIn

O método fadeIn é um método que serve para fazer elementos que estão escondidos aparecerem, transitando do transparente para o visível. Observe um exemplo de sua chamada no código a seguir (Figura 11.24):

```
1  <script>
2
3  $(function(){
4  $('#box').hide();
5
6  $('#box').fadeIn();
7  });
8
9  </script>
10
```

Figura 11.24. Exemplo da chamada do método fadeIn() no código jquery.

Nesse exemplo, inicialmente escondemos o elemento usando o comando "hide()" e depois fizemos ele aparecer usando o ".fadeIn();". O método fadeIn possui uma série de atributos que mudam detalhes de sua execução. Para mais detalhes sobre o método consulte a sua documentação.

11.10.2. FadeOut

O método fadeOut é o método oposto ao método fadeIn. Ele faz com que os elementos sumam, transitando do visível para o transparente. Observe um exemplo de sua chamada no código a seguir na figura 11.25:

```
1  <script>
2
3  $(function(){
4  $('#box').show();
5
6  $('#box').fadeOut(3000);
7  });
8
9  </script>
10
```

Figura 11.25. Exemplo da chamada do método fadeIn() no código jquery.

Nesse exemplo inicialmente fizemos o elemento aparecer (para não haver possibilidade de este estar escondido via css, por exemplo) usando o comando "show()" e depois fizemos ele desaparecer usando o ".fadeOut();". No exemplo passamos o número "3000" como parâmetro, esse valor é um parâmetro que muda a execução do fadeOut fazendo com que o tempo de duração para que o elemento se esconda seja de três segundos. Essa passagem de atributos também está disponível para o método "fadeIn". Para mais detalhes sobre o método consulte a sua documentação.

11.10.3. Combinando as animações de fade

A seguir, na figura 11.26, observe um código onde fazemos o elemento aparecer e desaparecer utilizando os métodos fadeIn e fadeOut:

```
<html>
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript"  src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
<script type="text/javascript">
$(function(){

    $('#apa').click(function(){

        $('#box').fadeIn(3000);
    });
    $('#des').click(function(){

        $('#box').fadeOut(500);
    });

});
</script>
<style type="text/css" media="all">
#box{
background:red;
width:300px;
height:300px;
}
</style>
</head>
<body>
<div id="box"></div>
<a href="#" id="apa">Click me!</a><a href="#" id="des">Click me!</a>
</body>
</html>

```

Figura 11.26. Exemplo de combinação dos métodos fadeIn() e fadeOut().

11.10.4. slideUp e slideDown

O método .slideUp serve para esconder elementos, mas diferetente do fadeOut, esse método esconde elementos fazendo com que esses deslizem para cima.

O método .slideDown é o método oposto ao slideUp. Ele faz elementos que estão invisíveis aparecer, como se estivessem deslizando para baixo. A seguir um exemplo do uso desses dois métodos combinados (Figura 27):

```
<html>
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript"  src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
<script type="text/javascript">
$(function(){

    $('#apa').click(function(){

        $('#box').slideUp();
    });
    $('#des').click(function(){

        $('#box').slideDown();
    });

});
</script>
<style type="text/css" media="all">
#box{
background:red;
width:300px;
height:300px;
}
</style>
</head>
<body>
<div id="box"></div>
<a href="#" id="apa">Click me!</a><a href="#" id="des">Click me!</a>
</body>
</html>

```

Figura 11.27. Exemplos da utilização dos métodos slideUp() e slideDown().

Os recursos de slide em muitas circunstâncias podem ser substituídos pelo uso do método slideToggle que foi explicado anteriormente.

11.10.5. Animate

Para fazer animações mais elaboradas o jquery disponibiliza o método ".animate". que possui uma série de atributos para customizar a animação. Nesse tópico iremos mostrar um exemplo simples do uso desse método, pois as possibilidades que ele oferece são muitas e cabe ao estudante que se interessar por esse ramo fazer um estudo mais aprofundado. No exemplo que iremos apresentar, um determinado elemento se movimenta da tela quando clicamos nos links. Visualize o exemplo apresentado na figura 11.28:

```
<html>
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript"  src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
<script type="text/javascript">
$(function(){

$('#go').click(function(){
$('#box').animate({
"left":"300px"
});
});
$('#back').click(
function(){
$('#box').animate({
"left":"0px"
});
}
);
});
</script>
<style type="text/css" media="all">
#box{
background:red;
width:300px;
height:300px;
position:relative;
}

</style>

</head>
<body>
<div id="box"></div>
<a id="go" href="#">Vai</a><a id="back" href="#">Volta</a>
</body>
</html>

```

Figura 11.28. Código que se utiliza do método animate para movimentar elementos.

No exemplo, para movimentarmos o elemento, usamos apenas o atributo “left”. Existem vários atributos que podem compor a animação, como por exemplo, a opacidade do elemento e a distância que o mesmo se encontra do topo. Um detalhe importante em relação a esse tipo de animação é o atributo css position que se refere à forma de posicionamento dos objetos na tela. A seguir, na figura 11.29, é mostrado o mesmo exemplo utilizando mais de um atributo:

```
<script>
$(function(){

$('#go').click(function(){
$('#box').animate({
"left":"300px","opacity":"0.25","width":"100px","height":"100px"
});
});
$('#back').click(
function(){
$('#box').animate({
"left":"0px","opacity":"1","width":"300px","height":"300px"
});
}
);
});
</script>

```

Figura 11.29. Exemplo com o método animate com mais de um atributo.

Apesar do uso do método .animate ser simples é desejado que o programador já possua uma certa experiência com javascript e css para que possa tirar mais proveito dessa tecnologia.

11.11. Usando o Ajax

11.11.1. O que é Ajax

A tecnologia Ajax (Asynchronous JavaScript and XML - JavaScript Assíncrono e XML) é uma forma de buscar dados em um arquivo externo a aplicação que pode estar na mesma pasta ou num servidor diferente. Antes do uso dessa tecnologia toda mudança significativa que o desenvolvedor queria fazer na sua aplicação web precisava recarregar toda a página, o que em muitos casos levava muito tempo. O Ajax permite que elementos tenham seus conteúdos e propriedades mudados em tempo de execução sem a necessidade de recarregar a página.

As primeiras aplicações que utilizavam essa tecnologia faziam isso utilizando comandos javascript e por muito tempo seu uso se tornava um problema, pois alguns navegadores implementavam esse recurso de formas diferentes e o fato de ter que programar de forma diferente para cada navegador, por muito tempo era uma dor de cabeça para desenvolvedores e empresas.

A biblioteca jquery tem em sua implementação suporte a utilização de requisições assíncronas e com a vantagem de não precisar criar uma versão diferente para cada navegador, sendo seu uso extremamente simplificado.

Para demonstrar o uso dessa tecnologia, a seguir será mostrado um exemplo no qual dados são pegos de forma assíncrona de uma página php. Nesse exemplo a página hospedada no servidor fará todo processamento dos dados enquanto a nossa aplicação somente irá mandar dados e exibir os resultados.

No exemplo proposto faremos uma página que busca resultados em um banco de dados. Faremos a página de busca utilizando apenas jquery. Nessa página haverá um formulário onde digitaremos o nome de uma pessoa e um botão para disparar a busca. Ao clicar no botão mandaremos um nome para que a página feita em php hospedada no servidor faça uma busca no banco de dados e retorne os resultados. Esse resultado, mesmo sendo mostrado em outra página, será carregado assincronamente na nossa página feita em jquery.

O banco de dados utilizado para o exemplo é o mysql e o comando de criação da tabela é o seguinte:

```
CREATE TABLE IF NOT EXISTS ‘pessoa’ (
  ‘id’ int(11) NOT NULL AUTO_INCREMENT,
  ‘nome’ varchar(255) NOT NULL,
  PRIMARY KEY (‘id’)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

O uso de banco de dados nesse exemplo foi uma opção para enriquecer o aprendizado, no entanto as requisições não precisam obrigatoriamente desse recurso. As requisições assíncronas podem ser feitas em páginas que somente processam os dados que são enviados e até mesmo em páginas estáticas, apenas com o intuito de trazer dados que estão armazenados.

Após a criação do banco criaremos o código php que rodará no servidor. Esse código receberá uma string vinda da página jquery pelo método *POST*, procurará no banco de dados por registros que possuam essa string no campo nome e simplesmente imprimirá o resultado. No exemplo utilizamos a linguagem php por ser uma linguagem popular, fácil de usar e livre, no entanto a utilização de requisições assíncronas podem ter como fonte páginas implementadas em qualquer linguagens de programação que tenha suporte a programação web.

O exemplo funcionará da seguinte forma: a página principal possui apenas código HTML e jquery. Ao clicarmos no botão “pesquisar” será disparada uma requisição assíncrona utilizando o método POST de envio. Os dados que serão enviados serão um nome e uma variável inteira. A variável inteira será uma forma de confirmar que a busca vem da página principal e a string terá um nome que será buscado no banco. A página secundária, encarregada de processar a operação, receberá o nome e fará uma pesquisa no banco de dados. Essa pesquisa será exibida na página principal no elemento alvo.

O método Ajax possui vários atributos que servem para customizar sua busca e atender as mais diversas necessidades. No exemplo utilizaremos apenas os essenciais para a execução. Observe na figura 11.30 o formato da chamada do método Ajax:

```
3
4  $.ajax({
5
6      url:'http://127.0.0.1/minicurso_jquery/pegarNome.php',
7      type:'POST',
8      data:'nome='+ nome + '&enviar=1',
9      success: function(result){
10
11 }
12 });
13
```

Figura 11.30. Chamada do método Ajax.

O atributo “url” armazena o caminho para o qual a requisição será enviada. Esse valor deve conter justamente a página para onde a informação é enviada e processada. No exemplo, a requisição é enviada para o servidor local, mas note que essa requisição pode ser enviada para servidores diferentes. Enviar uma requisição para um servidor diferente é um recurso muito útil tendo em vista que usando esse recurso o desenvolvedor pode dividir o processamento em servidores diferentes.

O atributo “type” informa ao método qual será a forma de passagem de informação. Existem dois métodos para o envio de informação na web: GET e POST. O método GET é mais rápido, no entanto pouco seguro, pois envia os dados para o servidor na url. O método POST envia seus dados de forma criptografada, o que torna uma opção segura em relação ao outro método.

O atributo “data” representa os dados que são enviados. Para enviar os dados é usado a seguinte notação “nomeDoDado=Valor”. Em casos que queremos passar mais de um valor utilizamos o símbolo “&” para conectar os dados. Dessa forma para passar dois valores o atributo dados teria esse formato “nomeDoDado1=Valor1& nomeDoDado2=Valor2”.

O atributo “success” guarda a função que é executada em caso de sucesso da requisição. Note que no exemplo a função tem um argumento de nome “result”. Esse argumento guarda tudo que foi retornado pela página de processamento. O result do nosso exemplo trará os nomes das pessoas cadastradas, mas esse retorno não precisa ser necessariamente um texto comum. O javascript tem uma vasta quantidade de métodos que manipulam arquivos XML e essa pode ser uma opção de formato para o retorno do servidor, permitindo assim que processamentos posteriores sejam feitos no retorno.

A seguir observe o código da página principal (Figura 11.31):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>

<style>
#resultado{
border: 3px solid;

}
</style>
</head>
<body>
<input type='text' id='nome'/><button id='botao'>Pesquisar</button>
<div id="resultado">
Faça a pesquisa!
</div>
<script src="jquery-1.7.1.min.js"></script>

<script>
$(function(){

$('#botao').click(
    function(){
        var nome = $('#nome').attr('value');

        $.ajax({

            url:'http://127.0.0.1/minicurso_jquery/pegarNome.php',
            type:'POST',
            data:'nome='+ nome + '&enviar=1',
            success: function(result){
                  $('#resultado').html(result);
            }
        });
    }
);

});
</script>
</body>
</html>
```

Figura 11.31. Código da página principal para envio dos dados.

No nosso exemplo, ao clicarmos no botão o texto que está contido no campo de texto é armazenado na variável nome e enviado para a página de processamento. Repare que a função que será executada em caso de sucesso simplesmente pega o conteúdo gerado na página de processamento e adiciona na div “resultado”. Todo esse processo ocorrerá sem que a página seja recarregada. Esse tipo de funcionalidade dá um aspecto contínuo à página tornando-a mais intuitiva para o usuário final. A seguir observe o código responsável pelo processamento (Figura 11.32):

```php
<?php
if (isset ($_POST['enviar'])) {

    if (isset($_POST['nome'])) {
        $nome = $_POST['nome'];

        //estabelecer a conexão
        $link = mysql_connect("localhost", "root", "") or die("Não foi possível conectar.");

        //selecionar o banco de dados
        mysql_select_db("teste") or die("Não foi possível selecionar o banco de dados.");

        //seleciona os registros com o nome igual ao nome passado por post
        $consulta = "SELECT * FROM pessoa WHERE nome like '%".$nome."%'";
        $resultado = mysql_query($consulta) or die("Erro na execução da consulta.");

        //salva os registros da consulta em $reg.
        while($reg = mysql_fetch_assoc($resultado)){//enquanto hoverem registros
            //imprimindo o resultado
            echo "Nome: ".$reg['nome']."<br/>";

        }
    }
    else
        echo "Valor não enviado!";
}
?>
```

Figura 11.32. Código php responsável pelo processamento dos dados recebidos.

Sendo assim, o processamento da busca acontece de forma paralela à execução da página e não há a necessidade de recarregar todo o conteúdo. Essa maneira de tratar os dados está cada vez mais popular entre os programadores web porque torna a página mais rápida e elegante. Como foi visto no exemplo, em poucas linhas o Ajax pode ser implementado utilizando a biblioteca jquery, sendo esse recurso um dos grandes diferenciais dessa biblioteca.

11.12 Conclusão

A utilização de jquery é um fenômeno crescente no desenvolvimento de aplicações web. A cada dia novos desenvolvedores aderem a esse grupo. A biblioteca fornece uma série de funcionalidades que facilitam o trabalho do programador, fazendo programas com menos linhas de código e com mais funcionalidades, contudo sem diminuir ou interferir na forma individual de programar de cada um.

A maneira como o código jquery é escrito permite uma separação bem definida sobre o que é estrutural e o que é relacionado a comportamento. O jquery é uma biblioteca que permite maior reuso e maior performace sem que haja preocupação com detalhes de implementação e sem perder a flexibilidade oferecida pelo javascript.

11.3. Bibliografia utilizada

BuiltWith.com (2011) “jQuery Usage Statistics”, http://trends.builtwith.com/javascript/jQuery, Março.

BuiltWith.com (2011) “Query Version and Plugin Usage Report”, http://blog.builtwith.com/2011/10/31/jquery-version-and-usage-report/, Março.

jqueryui.com (2010) “jQuery user interface”, http://jqueryui.com/, Março.

MURPHEY, R. (2010) “Fundamentos de jQuery”, http://jqfundamentals.com/book/index.html, Março.

SILVA, M. S. (2008) “jQuery: a biblioteca do programador JavaScript”, Novatec - Editora, São Paulo, ISBN 978-85-7522-178-5, p.16-20.

Capítulo 12

Descoberta de Informação Através da Mineração de Texto – Fundamentos e Aplicações

José Alberto Sousa Torres

Abstract

This chapter provides a Text Mining overview, presents its fundamentals, architecture, key processes and techniques used to automaticaly process text documents in order to obtain useful and structured information. We also show the main aspects of text mining in practice using the Google as example.

Resumo

Este capítulo fornece uma visão geral da área de Mineração de Textos, apresenta os fundamentos, arquitetura, principais processos e técnicas utilizadas no processamento computacional de documentos de texto com o objetivo de obter informação útil e estruturada. Serão apresentados, ainda, os principais aspectos da mineração de textos aplicados em uma situação real, utilizando o Google como exemplo.

12.1. Introdução

O crescimento exponencial da quantidade de documentos publicados na internet é uma realidade que se estende desde a década passada. Não há uma estatística oficial da quantidade de páginas existentes na web, o tamanho atual é estimado, geralmente, a partir da quantidade de páginas indexadas pelos principais mecanismos de busca.

Um levantamento realizado pela Universidade de Tilburg estimou que em fevereiro de 2012 já existiam mais de 50 bilhões de páginas indexadas pelo Google[8]. A última estatística emitida pelo próprio Google em relação ao tamanho de sua base foi publicada no Blog oficial da companhia[9] em julho de 2008, à época, a empresa informava ter alcançado a marca de 1 trilhão de endereços únicos indexados.

[8] http://www.worldwidewebsize.com/ acessado em 07/03/2012 às 12:20

[9] http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html acessado em 01/03/2012 às 11:30

A grande questão quando lidamos com quantidade tão grande de documentos é como localizar de forma rápida e precisa uma informação específica. Seguramente, podemos afirmar que, sem o auxílio das atuais ferramentas de busca, a tarefa de encontrar a informação de forma eficiente e eficaz não seria possível. Se levarmos em conta que muitas destas páginas possuem documentos de texto anexados, como arquivos em PDF, RTF, DOC, ODS, dentre uma infinidade de outros formatos, o problema se tornaria ainda maior. É neste contexto que popularizou a mineração de texto, que pode ser genericamente definida como um processo que busca extrair informação útil de coleções de documentos de texto através da identificação e exploração de padrões.

A mineração de texto sofreu forte influência da mineração de dados (*data mining*), razão pela qual não causa surpresa o fato de que são encontradas similaridades nas arquiteturas em alto nível de ambas. A principal diferença é o fato que a mineração de dados utiliza informações que se encontram em formato estruturado, conquanto boa parte do esforço despendido na mineração de textos tem como objetivo a construção de um modelo de dados estruturado, a partir de documentos não estruturados, escritos em linguagem natural.

12.2. Arquitetura

De uma forma geral, a arquitetura das aplicações de mineração de texto pode ser definida em função de três atividades, o pré-processamento, a etapa da aplicação da técnica principal – aqui chamada de mineração, e a última fase, de apresentação dos resultados e avaliação (Figura 12.1).

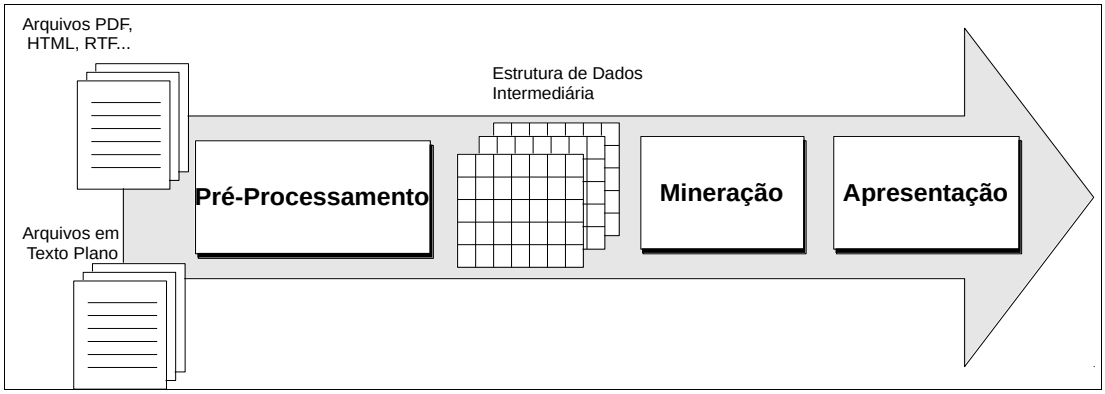


Figura 12.2. Arquitetura do Processo de Mineração de Texto

Resumidamente, o processo se inicia com definição dos documentos que irão compor o conjunto de entrada. Estes documentos passam por uma etapa de pré-processamento que vai transformar os dados de entrada em um conjunto intermediário, semi-estruturado, que será a base para a aplicação das principais técnicas de mineração de texto. Por fim, o resultado da aplicação da técnica é apresentado ao usuário para que a validade da informação seja avaliada.

12.2.1. Documentos

Antes de iniciar o detalhamento de cada uma das atividades da mineração de textos, convém descrever a peça chave de todo o processo – os documentos de texto. O conjunto de documentos textuais em linguagem natural compõe a entrada do processo de mineração. São documentos escritos nos mais diversos idiomas e nos mais diferentes formatos.

Com base no formato destes arquivos de texto, foram definidas duas categorias de documentos, os não estruturados e os fracamente estruturados. Os textos fracamente estruturados são definidos como aqueles que possuem marcas de formatação em sua composição - a exemplo do negrito e itálico, tamanhos diferentes para as letras e uma série de marcas de estilo. Alguns exemplos de documentos fracamente estruturados são os formatos HTML e PDF. Os documentos não estruturados são aqueles escritos em texto plano, sem qualquer marca de formatação. Outra linha de estudo considera que a simples presença dos sinais de pontuação, a utilização de palavras com iniciais em maiúsculo e o uso de caracteres especiais em um documento já são indicativos de uma estruturação fraca, já que auxiliam na identificação de importantes componentes do texto, como parágrafos, títulos e cabeçalhos. Estes mesmos autores classificam os documentos HTML, PDF como documentos semi-estruturados, e os demais como fracamente estruturados.

É interessante observar, entretanto, que os conceitos de documento estruturado ou não estruturado apresentados dizem respeito unicamente a análise automática, a uma operação pautada no processamento do texto por uma máquina. Visualmente, uma página HTML ou um documento PDF possuem uma estruturação de tópicos, seções e uma série de elementos que facilitam o entendimento humano do documento, de forma que, para a nossa leitura, estes documentos são claramente estruturados.

Os documentos de texto são a peça chave, mas é o conjunto destes que comumente é utilizado como entrada nos processos de mineração. É fácil encontrar conjuntos de entrada com centenas, milhares e até milhões de elementos. Sobre estes grupos de documentos é que são aplicadas as técnicas, obtidos os padrões, extraídas as informações úteis, mapeamentos e relacionamentos. Algumas das aplicações de mineração de texto, como as de recuperação de informação ou identificação de linguagem, trarão o resultado esperado quando utilizado conjuntos de entrada formados por apenas um documento, mas aplicações deste tipo vão de encontro a um dos principais agentes motivadores do avanço desta área de estudo, a necessidade de se obter informação de forma eficiente e eficaz, atuando sobre grandes massas de dados. Não é interessante desenvolver aplicações onde o esforço para construir a ferramenta será, provavelmente, maior do que o esforço necessário para a análise manual do documento.

12.2.2. Pré-Processamento

O principal objetivo da etapa de pré-processamento é transformar o texto de entrada em um formato intermediário, estruturado, que possa ser processado automaticamente por máquinas. Em processos de mineração, este formato intermediário é usualmente representado como um vetor de termos com atributos associados. A representação deste vetor de termos, denominado nas publicações em inglês de *bag-of-words*, pode ser observada no Quadro 12.1, onde *d* representa o documento, *t* o conjunto de termos extraídos do texto e *a* os atributos associados.

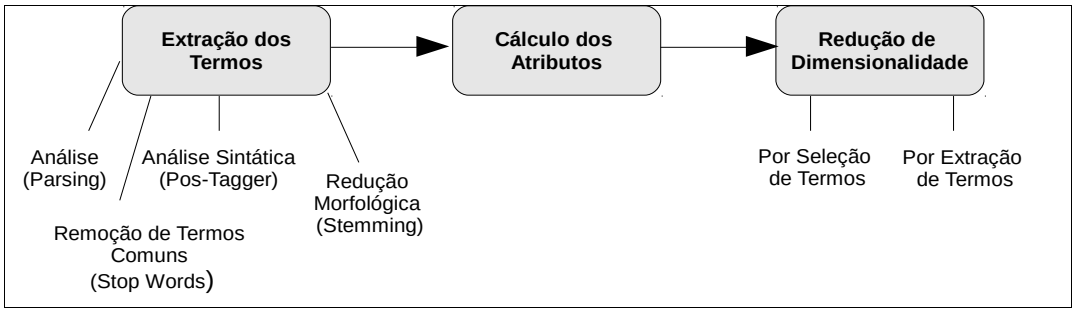
Quadro 12.1. Representação do texto em função dos termos e de seus atributos

$d_1 = (t_1a_1,a_2, \ldots a_n,, t_2a_1,a_2, \ldots a_n , \ldots t_na_1,a_2, \ldots a_n)$

A definição do que é um “termo” (*t*) vai depender do objetivo da aplicação que está sendo desenvolvida, já que os documentos podem ser quebrados em seções, parágrafos, sentenças, palavras e até sílabas. A definição mais encontrada nos trabalhos de mineração de textos é a de termo como o conjunto de palavras existentes. Associados aos termos, é definido um conjunto de atributos para representar algum tipo de característica do termo a que está vinculado. O tipo de atributo é dependente da granularidade em que foi definido o termo, portanto, recomenda-se definir os atributos farão parte do vetor só depois da decisão de como o texto será segmentado.

Não existe na literatura um padrão de procedimentos a ser executado na etapa de Pré-processamento do texto, entretanto, um conjunto de atividades recorrentemente encontradas em trabalhos da área foi compilado e é apresentado na Figura 1.2. Cada uma destas atividades será detalhada nas seções subseqüentes.

Figura 12.2. Atividades de Pré-Processamento



12.2.2.1. Extração dos Termos

A atividade de extração dos termos tem como principal objetivo definir o conjunto de termos que irá compor o vetor de termos (*bag-of-words*). Nesta etapa o texto original é quebrado em partes menores, a depender da granularidade selecionada, e um processamento simplificado é realizado visando melhorar os índices de acerto ao final do processo completo de mineração.

Das atividades vinculadas à extração de termos, apenas a Análise (*parsing*) é obrigatória para que o processo de mineração de texto possa ser executado, já que esta é a etapa responsável pela fragmentação do texto original. As demais atividades são auxiliares e podem ou não ser executadas, a depender do tipo de problema que está sendo tratado.

a) Análise (*Parsing*)

A etapa de análise visa fragmentar o texto original com base no conceito de "termo" adotado. Além deste procedimento, é realizada a remoção das marcações e normalização da estrutura dos documentos fracamente estruturados, a exemplo dos formatos HTML, RTF ou PDF.

A depender do objetivo da aplicação que está sendo desenvolvida, a informação ligada a formatação do texto, como negrito, sublinhado, tamanhos de letra e uma série de outros atributos, podem ser descartados ou armazenados em um modelo diferente de dados. Em uma aplicação para localização e indexação de notícias encontradas na internet, por exemplo, normalmente, ir-se-á encontrar o título da notícia em um tamanho de fonte

superior ao encontrado no restante do artigo. Neste contexto, a formatação se torna fator importante na tarefa de identificar um item significativo para o processo de mineração do texto – o título da notícia, devendo esta informação ser mantida para ser utilizada no restante do processo.

Imagine, por outro lado, que uma aplicação foi construída com o objetivo de identificar automaticamente o idioma em que está escrito o texto de uma coleção de documentos. Neste caso, as marcações e o estilo aplicados ao texto não trazem qualquer benefício ao processo devendo, portanto, ser descartados.

Por padrão, na saída desta etapa é criada a representação inicial da *bag-of-words,* e são preenchidos os espaços correspondentes aos termos, sejam as palavras, as sentenças ou qualquer outro nível de fragmentação definido para o texto.

b) Remoção de Termos Comuns (*Stop Words*)

"*Stop Words*" é uma expressão da língua inglesa que foi cunhada, no âmbito do processamento automático de textos, com o objetivo de identificar um grupo de palavras que não representam informações relevantes do texto e nem auxiliam na identificação destas informações.

Estas palavras aparecem constantemente na quase totalidade dos textos escritos, independentemente do contexto ou área do conhecimento a que pertencem. São termos comuns e que possuem papel de ligação entre as palavras principais das sentenças. Os principais exemplos de *stop words* são os artigos, as preposições e os pronomes, dentre outras classes correlatas.

A remoção de termos comuns é utilizada quando as palavras que compõe o texto são definidas como os termos do vetor. A principal vantagem em promover a exclusão é a redução da necessidade de processamento gerada pela considerável diminuição do tamanho do vetor de termos.

A principal estratégia para tratamento e remoção dos termos comuns é a utilização de um banco de dados de palavras, organizadas por língua. No Quadro 12.2 podemos observar uma relação de termos comuns encontrados na língua portuguesa, já no Quadro 12.3 observamos alguns termos comuns da língua inglesa.

Quadro 12.2. Exemplos de termos comuns (*stop words*) na língua portuguesa

à, ao, aos, aquela, aquelas, aquele, aqueles, aquilo, as, às, até, como, da, daquela, daquelas,daquele, daqueles, daquilo, das, de, dela, delas, dele, deles, depois, dos, é, ela, elas, ele, eles, em, entre, era, era, eram, éramos, essa, essas, esse, esses, esta, está, estamos...

Quadro 12.3. Exemplos de termos comuns (*stop words*) na língua inglesa

able, about, above, according, accordingly, across, actually, after,

afterwards, again, against, all, allow, allows, almost, alone, along, already, also, although, always, am, among, amongst, and, another, any, anybody, anyhow, anyone, anything, anyway, been, before, beforehand, behind, would, yes, yet, you, your, yours, yourselves...

Os termos comuns, entretanto, não se mostram inapropriados em todas as tarefas de mineração de textos. Por serem extremamente dependentes da língua em que o texto foi escrito, diversas aplicações utilizam as listas de *"stop words"* como ferramenta para identificação automática do idioma em que o documento foi escrito.

c) Análise Sintática (*POS Tagger*)

O principal objetivo da Análise Sintática é definir o tipo gramatical dos termos presentes no vetor de termos. Como a classificação gramatical é vinculada as palavras, assim com ocorreu na etapa de remoção dos termos comuns, esta técnica é mais comumente utilizada em aplicações onde o texto foi fragmentado em função de suas palavras.

A principal dificuldade desta técnica é o fato de que não é raro, em linguagens naturais, que uma mesma palavra possua diferentes classes gramaticais a depender da posição que ocupe na sentença. Um exemplo desta ambiguidade é a palavra "prova", que na frase "*A sua atitude prova o seu caráter*" é classificada como um verbo, mas em "*A prova estava difícil*" é definida como um substantivo. O tratamento computacional de ambiguidades é uma tarefa extremamente complexa de ser resolvida. Apesar dos principais algoritmos de análise sintática atingirem níveis de acerto superiores a 97%, quando são tratadas apenas as questões ambíguas, o grau de acerto é pouco superior a 50% (Manning, 2011).

São diversas as estratégias desenvolvidas para identificar a classe das palavras do texto. O ponto em comum da maioria dos métodos é a utilização de um corpus anotado de textos como base de conhecimento para o processo de classificação das novas palavras. Um corpus anotado, neste contexto, é coleções de documentos onde a classificação gramatical de cada uma das palavras foi manualmente definida e explicitamente marcada no texto.

Para auxiliar o desenvolvimento de aplicações, diversos corpora foram construídos e disponibilizados na rede. O corpus de Brown (FRANCIS; KUCERA, 2012), por exemplo, é uma das principais referências na língua inglesa, foi compilado na década de 1960 com cerca de 500 textos e é composto por 1.014.312 palavras, classificadas em 87 diferentes categorias gramaticais. O Penn Treebank (MARCUS; SANTORINI; MARCINKIEWICZ, 1993) é um corpus mais recente, composto por cerca de 4,5 milhões de palavras na língua inglesa, classificadas em 48 diferentes categorias gramaticais, apresentadas na Tabela 12.1

Tabela 12.1. Marcações gramaticais utilizadas no Corpus Penn Treebank

Tag	Descrição	Tag	Descrição
CC	Coordinating	TO	*To*

CD	Cardinal number	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential - *there*	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund/present
IN	Preposition/subordinating participle /conjunction	VBN	Verb, past participle
JJ	Adjective	VBP	Verb, non-3rd ps. sing. Present
JJR	Adjective, comparative	VBZ	Verb, 3rd ps. sing. Present
JJS	Adjective, superlative	WDT	wh-determiner
LS	List item marker	WP	wh-pronoun
MD	Modal	WP$	Possessive wh-pronoun
NN	Noun, singular or mass	WRB	wh-adverb
NNS	Noun, plural	#	Pound sign
NNP	Proper noun, singular	$	Dollar sign
NNPS	Proper noun, plural	.	Sentence-final punctuation
PDT	Predeterminer	,	Comma
POS	Possessive ending	:	Colon, semi-colon
PRP	Personal pronoun	(	Left bracket character
PP$	Possessive pronoun	)	Right bracket character
RB	Adverb	"	Straight double quote
RBR	Adverb, comparative	'	Left open single quote
RBS	Adverb, superlative	"	Left open double quote
RP	Particle	'	Right close single quote
SYM	Symbol (mathematical or scientific)	"	Right close double quote

Em português, um dos corpus mais conhecidos é o Mac-Morpho (MARCHI,2003) , formado por artigos jornalísticos retirados da Folha de São Paulo no ano de 1994. Este corpus possui 1.167.183 palavras em sua composição e é mantido pelo projeto LacioWeb[10].

O uso dos corpora são peças chave para a implantação das principais técnicas de marcação automática das categorias gramaticais das palavras. Estas coleções anotadas de texto funcionam como treinamento para os principais métodos utilizados, sejam estatísticos, como o modelo oculto de Markov, ou baseados em aprendizado de máquina, como os algoritmos de Máquinas de Vetores de Suporte, algoritmo do Vizinho Mais próximo ou Redes Neurais.

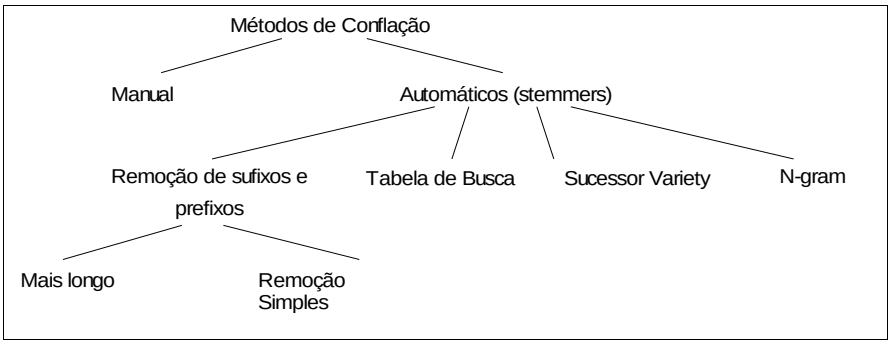
d) Redução Morfológica (*Stemming*)

[10] http://www.nilc.icmc.usp.br/lacioweb/corpora.htm acessado em 28/02/2012 às 11:30

O "*Stemming*" é um método de padronização de palavras amplamente utilizado, desenhado para permitir a redução de termos à menor unidade semântica possível (PAICE, 1990). A definição é que uma palavra típica possui um radical ("*stem*") que se refere a alguma idéia central, e que certos prefixos e sufixos são utilizados para modificar o seu significado ou adequar a palavra a uma determinada regra sintática. O "*stemming*" promove a remoção destes prefixos e sufixos de modo a reduzir a palavra a sua "essência". Estudo de Hull (1996) comprovou que a utilização dos algoritmos de redução morfológica pode gerar uma melhora no desempenho do processamento em aproximadamente 5%.

Os algoritmos de redução morfológica possuem características próprias que os levam a uma separação por classes. Frakes & Baeza-Yates (1992) propuseram uma taxonomia para classificação, exibida na Figura 12.3.

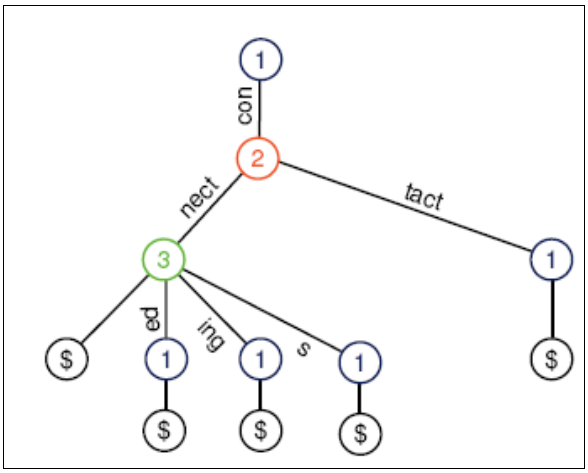
Figura 12.3. Taxonomia para Algoritmos de Stemming



O algoritmo básico de tabela de busca se utiliza de uma tabela de termos indexados e radicais. O princípio de funcionamento destes algoritmos é a consulta a esta estrutura de dados. Um algoritmo de tabela de busca que utiliza análise morfológica foi proposto por Krovetz (1993) – o K-Stemmer. A cada operação efetuada pelo algoritmo é realizada uma consulta a um dicionário (tabela de busca) de forma a determinar se o radical ("stem") encontrado é válido. A etapa de análise derivacional do algoritmo também realiza uma consulta numa lista dos principais sufixos derivacionais.

O "*sucessor variety*" é um outro tipo de algoritmo, e tenta, após uma rebuscada análise da frequência e tamanho dos prefixos ao longo das variações morfológicas de uma palavra, identificar heuristicamente seu potencial radical. Stein & Potthast (2007) definiram um método baseado no *Sucessor Variety* em que propõe a utilização de uma árvore de sufixos, onde os nós da árvore armazenam a informação sobre a freqüência. Um exemplo desta árvore para as palavras "connect", "connected", "connecting", "connects" e "contact" pode ser visualizado na Figura 12.4. O caractere "$" indica o fim de uma sequência de caracteres.

Figura 12.4. Exemplo de árvore de sufixos (Stein e Potthast, 2007).



Já nos algoritmos baseado em n-gram, medidas de associação são calculadas entre pares únicos de letras consecutivas, chamados de diagramas. A medida de similaridade é calculada a partir da identificação dos diagramas em comum entre pares de palavras. O valor da similaridade é calculado de acordo com a equação do Quadro 12.4. As medidas de similaridade são determinadas para todos os pares de termos no banco de dados, formando uma matriz de similaridade. Uma vez disponível a matriz, os termos são agrupados utilizando-se um método de aglomeração de ligação simples (*single link clustering method*) como o descrito por Croft (1977).

Quadro 12.4. Equação para cálculo de similaridade

$$S = 2 \times C / A + B$$

onde A é o número de diagramas únicos na primeira palavra;
onde B é o número de diagramas únicos na segunda palavra;
onde C é o número de diagramas únicos compartilhados por A e B.

A análise e a avaliação da performance de um método de *stemming* baseado na abordagem de N-gram foi realizada por Kosinov (2001). O estudo apontou que a técnica obteve resultados superiores, em alguns casos, aos obtidos com um algoritmo baseado em remoção de sufixo. Outro ponto interessante sobre o algoritmo apresentado é o fato de que a técnica N-gram é independente de linguagem.

A última classe de algoritmos de redução morfológica é a de remoção de sufixos. Esta é a técnica mais largamente utilizada, tendo como principal destaque o método proposto por Porter (1980). A base do método é a utilização de uma lista específica de terminais que são removidos das palavras a partir de determinados critérios, de forma a deixar apenas a "raiz" da palavra.

Os algoritmos de *stemming* baseados na remoção dos sufixos não se preocupam com a estrutura de formação nem a semântica das palavras. Em grande parte das conversões, após a remoção dos sufixos, o resultado traz uma seqüência de letras sem significação semântica, com função meramente estatística. No algoritmo de Porter, os sufixos são removidos simplesmente para prover uma melhoria no desempenho dos

sistemas de recuperação da informação e mineração de dados, e não como um exercício linguístico, ou seja, não são sempre óbvias as circunstâncias em que um sufixo deve ser removido.

De uma forma geral, são fornecidas uma lista de sufixos e outra lista de condições de remoção. Quando a condição é satisfeita a operação sobre a palavra é realizada. Quando a palavra é muito curta, não é realizada a remoção do sufixo, mesmo que exista uma condição que indique a operação. O tamanho da palavra é dado pela contagem das letras, não havendo uma base linguística para esta observação. Sufixos complexos são removidos em diferentes etapas.

12.2.2.2. Cálculo dos Atributos

A definição e cálculo dos atributos dos termos é a atividade mais importante da etapa de pré-processamento do texto. Selecionar atributos que representem características pertinentes à aplicação que está sendo desenvolvida é essencial para que a etapa de mineração traga resultados satisfatórios.

Os atributos normalmente são definidos em função de dados estatísticos, estruturais ou lingüísticos. Os primeiros se relacionam a valores numéricos obtidos com base em informações estatísticas do texto em análise; os estruturais se baseiam na estrutura de organização do texto, é vinculado à posição do termo no documento; por fim, os lingüísticos se referem a informações voltadas às análises morfológica, sintática e semântica relacionadas a língua em que o texto foi escrito. Nas seções subseqüentes apresentaremos alguns atributos típicos utilizados em aplicações de mineração de texto.

a) Freqüência dos Termos (TF)

Esta métrica estatística leva em consideração apenas a frequência das palavras no texto. A quantidade de vezes que a palavra é encontrada no texto de entrada é registrada e associada a uma das dimensões do vetor de termos.

Uma variação desta métrica foi proposta por Luhn (1958) com o intuito de medir o grau de importância de uma palavra na caracterização da sentença a que pertence, a partir da frequência em que aparece no texto. Esta adequação permite que este atributo seja aplicado às sentenças do texto, e não apenas as palavras.

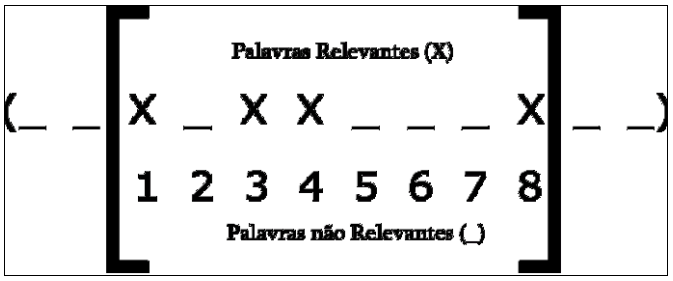
A hipótese é que um escritor normalmente repete certas palavras à medida que ele avança em sua argumentação ou quando redige um aspecto sobre determinado assunto. Ainda segundo o autor, é muito pouco provável que, em textos técnicos, uma palavra seja utilizada com diferentes significados ou que se utilize diferentes palavras para representar um mesmo conceito. Se por questões estéticas o autor do estudo faz um esforço significativo em busca de sinônimos, como há quantidade limitada de palavras com sentido correlato, chega um momento em que ele fica sem alternativas e acaba por repetir as palavras-chave.

É importante observar, entretanto, que a proposição “toda palavra com alta frequência é uma palavra-chave” não é verdadeira. Palavras muito comuns, geralmente,

possuem baixa importância, representando uma espécie de ruído do sistema. Palavras como artigos, preposições ou conectivos são encontradas em grande quantidade nos mais diversos textos, independente do tema que tratem, e não são importantes para a classificação destes documentos. Estas palavras, conhecidas como "*stop-words*", foram abordadas anteriormente.

O cálculo da frequência das palavras-chave para uma sentença é derivado da análise das palavras que a compõe. São dois aspectos a serem analisados: o cálculo do número de ocorrências das palavras que compõem a sentença e o cálculo do fator que representa a distância entre as palavras-chave. O cálculo da frequência é derivado da soma da número de ocorrências das palavras-chave encontradas na sentença (algumas das palavras mais frequentes são encontradas em conjunto umas com as outras dentro de uma sentença). Já em relação ao fator da distância entre as palavras-chave, quanto maior o número de palavras com grande quantidade de ocorrências encontradas fisicamente próximas uma das outras, mais representativa é esta informação para o artigo. Na Figura 12.5, os traços (*underline*) representam as palavras não relevantes, e o "*X*" representa as palavras relevantes de uma sentença. O primeiro passo para o cálculo da distância entre as palavras é diminuir o tamanho da sentença para a posição das palavras-chave (relevantes) mais periféricas. A sentença reduzida é representada na figura pelas palavras no interior dos colchetes.

Figura 12.5. Cálculo da distância das palavras de uma sentença



Neste ponto, o fator de importância é calculado elevando-se ao quadrado a quantidade de palavras relevantes (4^2). Em seguida, deve-se dividir este valor pelo número total de palavras dentro dos colchetes (8), obtendo como resultado o número 2 (16/8).

b) Localização da Sentença

Este atributo estrutural, proposto inicialmente por Baxendale (1958), é utilizado como atributo em aplicações onde as sentenças são definidas como termos. O cálculo do atributo se baseia na idéia de que as sentenças que ocorrem abaixo de certos títulos são positivamente relevantes e que as sentenças relevantes tendem a acontecer muito cedo ou muito tarde em um documento e em seus parágrafos. Desta forma, a primeira e a última sentença de um parágrafo ou do texto podem, por conseguinte, expressar a idéia principal do documento, devendo ser marcadas como sentenças chave do texto.

Em seu experimento, Baxendale observou que, em uma amostra de 200 parágrafos, a sentença principal encontrava-se na primeira posição em 85% dos casos, e na última posição do parágrafo em 7% dos casos. Nos 8% restantes, as informações relevantes encontravam-se entre a primeira e a última posição de um mesmo parágrafo. Apesar do valor pouco significativo, o autor considerou a inclusão da última sentença do parágrafo para manter a coesão textual, pois esta representava o elo com a primeira sentença do parágrafo posterior (MARTINS, 2001).

Na operacionalização do método, cada sentença recebe um peso de acordo com a sua presença no início ou no fim do parágrafo, no início ou fim do documento, ou imediatamente abaixo de um título de seção. Por fim, a peso da sentença é incrementado se esta aparece abaixo de certos títulos ou subtítulos de seção como, por exemplo, "resultados" ou "conclusão".

Segundo INDERJEET, GATES, B. e BLOEDORN (1999), este atributo é dependente de contexto. A estratégia de se obter o primeiro parágrafo, por exemplo, funciona apenas em documentos do gênero notícias e não necessariamente em todos os outros. Nenhuma técnica automática foi elaborada para definir o posicionamento ótimo das sentenças em outros domínios.

c) Palavras-chave no Título

A base do cálculo deste atributo, proposto por Edmundson (1969), é o fato de que as palavras constantes nos títulos e subtítulos do documento e de suas seções, retiradas as "*stop words*", são as que melhor representam o tópico principal do texto.

O valor deste atributo é a soma das vezes que a palavra/termo é encontrado nos títulos, subtítulos e seções do texto A operacionalização este atributo, depende da identificação dos elementos do texto durante etapa anterior ao cálculo dos atributos.

d) Palavra Temática *(Thematic Word Feature*)

Este atributo tenta identificar as palavras-chave do texto – denominadas palavras temáticas, que seriam marcadas no vetor de termos com valor correspondente. Para casos onde sentenças, parágrafos ou elementos de maior granularidade são definidos como "termos", pode-se definir o valor do atributo como sendo o valor da soma das palavras que constituem cada uma destas estruturas. Para cada palavra é calculado um peso a partir de uma função específica. Ao final, um percentual das palavras com maior peso é escolhido para compor o conjunto de palavras temáticas.

Diversas são as funções utilizadas para cálculo do peso das palavras. Uma das mais conhecidas e utilizadas é o quociente da Frequência do Termo (TF) pela Frequência Inversa do Documento (IDF), definida por Teufel e Moens (1997). A fórmula para cálculo é exibida no Quadro 12.5.

Quadro 12.5. Fórmula para Cálculo do TF-IDF

$$score(w) - f_{loc} * \log\left(\frac{100 * N}{f_{glob}}\right)$$

f_{loc}: frequência da palavra w no documento
f_{glob}: número de documentos que possuem a palavra w
N: número de documentos na coleção

Outra função foi proposta por Pardo (2003), que a definiu como Frequência Inversa da Sentença - TF-ISF. Como o próprio nome diz, esta é uma métrica aplicada a problemas onde as sentenças ou estruturas superiores, como parágrafos ou seções, são selecionadas como conceito de "termo".

Para o cálculo da medida TF-ISF são utilizadas a medida da frequência da palavra na sentença – F(w), o número de palavras na sentença a que a palavra pertence – n, e o número de sentenças em que a palavra aparece – S(w), conforme exibido no Quadro 12.6 (LAROCCA NETO et al. 2000).

Quadro 12.6. Fórmula para Cálculo do TF-ISF

$$TF-ISF(w) = F(w) * \frac{\log(n)}{S(w)}$$

e) Presença de Palavras da "Gist Sentence"

O cálculo do peso proposto por Pardo (2002) se utiliza dos métodos de frequência das palavras (Tchemra, 2009) e do TS-ISF (Larocca Neto et al. 2000) para definir a sentença com maior pontuação do texto, a qual, ao menos teoricamente, representaria a idéia principal do texto. Esta sentença é denominada *sentença-gist* do texto. A partir deste ponto são selecionadas todas as sentenças do texto que:

(1) possuam pelo menos uma das formas canônicas[11] que compõe a *sentença-gist*;

(2) possuam uma pontuação maior do que a média da pontuação de todas as sentenças do texto.

Uma variante deste método pode permitir que este atributo possa ser aplicado também às palavras. Para tanto, basta criar uma marcação para palavras encontradas no texto e que sejam encontradas, mesmo que em sua forma canônica, na *gist-sentence.*

f) Inicial Maiúscula (*UppercaseWord Feature*)

Segundo Kupiec (1995), nomes próprios são frequentemente importantes para definição das sentenças relevantes. O passo inicial para utilização deste método é a criação de uma lista de palavras iniciadas com letra maiúscula que não iniciem sentenças, que ocorram diversas vezes e que não sejam unidades de medida ou abreviaturas. O cálculo dos pesos das sentenças é feito com base na quantidade de palavras constituintes que se encontram na lista de palavras previamente definida, sendo que as sentenças em que uma palavra da lista

[11] Para alcançar a forma canônica de uma palavra, basta reduzir os verbos para o infinitivo e os substantivos e adjetivos para o masculino singular

foi encontrada pela primeira vez tem peso dobrado em relação às outras sentenças que possuem esta mesma palavra.

g) Linha Base (*baseline*)

O atributo da linha base foi definido por Lin (1999) e estabelece que o peso de cada sentença é baseado na sua posição no texto como um todo, sendo que a primeira sentença possui o maior valor e a última o menor valor.

h) Conectividade Léxica (*average lexical connectivity*)

O cálculo deste atributo, também proposto por Lin (1999), pressupõe que a sentença que compartilha muitos termos com outras sentenças é importante. Desta forma, define o peso da sentença com base no número de termos compartilhados entre esta e as demais, dividido pelo número total de sentenças do texto.

i) Tamanho do Termo

O cálculo deste atributo parte da hipótese de que as sentenças mais longas normalmente apresentam maior conteúdo informativo sendo, portanto, mais relevantes (KUPIEC et al., 1995). Às sentenças são atribuídos pesos proporcionais ao tamanho, sendo definido um valor prévio de corte (5 palavras, por exemplo).

Na abordagem conexionista de Pardo (2003), as sentenças são enquadradas em uma escala com 4 patamares, onde são considerados o tamanho médio das sentenças do texto e o tamanho da maior sentença do texto.

O tamanho das palavras em função do número de letras, e de parágrafos em função do número de palavras ou sentenças podem, por exemplo, ser utilizados como variações deste atributo.

j) Palavras Sinalizadoras (*cue method*)

Este atributo baseia-se na hipótese de que a provável relevância de uma sentença é afetada pela presença de determinadas palavras, consideradas relevantes no domínio do texto (*cue words*). O cálculo utiliza um dicionário de *cue words*, com palavras previamente selecionadas e divididas em três grupos distintos: as palavras bônus (*bonus words*), que representam as que são positivamente relevantes; as palavras stigma (*stigma words*) – que representam as negativamente relevantes; e as palavras nulas (*null words*), que são irrelevantes e devem ser ignoradas na contagem da frequência. Pesos negativos são atribuídos a cada palavra das sentenças que conste entre as palavras *stigma*, enquanto pesos positivos são atribuídos a cada palavra que conste na lista de palavras bônus. As palavras nulas são ignoradas pelo processamento. O peso final da sentença é formado pela média ponderada entre valores negativos e positivos de suas palavras constituintes.

Os dicionários são dependentes do contexto em que se encaixam as sentenças a serem analisadas. Em textos científicos, palavras como “método” ou “resultado” serão, muito provavelmente, altamente significativas, devendo constar no dicionário de palavras

bônus. Se utilizássemos o domínio notícias de jornal, por exemplo, estas palavras não seriam relevantes.

O método permite variações, como a utilização de frases ao invés de palavras. Quando utilizadas frases, estas são tipicamente curtas e obtidas de vocabulário de documentos relacionados a determinado contexto (HUI, 2002).

Teufel e Moens (1997) definiram, a partir da observação de sentenças chave em um determinado corpus, um dicionário de palavras com diferentes pesos. A classe de bônus foi subdividida em três diferentes grupos, com pesos distintos, enquanto as classes de palavras nulas e de palavras stigma foram mantidas em apenas um grupo. Parte da lista definida pode ser visualizada na Tabela 12.2.

Tabela 12.2. Dicionário de Palavras

Dicionário				
Palavras Stigma peso -1	**Palavras nulas**	**Palavras Bônus**		
		frases menos significantes – peso 1	*frases de valor intermediário – peso 2*	*frases mais significantes – peso 3*

for example
example
comments on an ...
comments on earlier drafts funded by
thankful
...
	// verbs
	am ...
	is
	. . .
	// articles
	the
	these
	those ...
		a central problem
		a comparison was made
		a growing need
		a limiting factor of
		. . .
		would be interesting
		little attention
			above show
			above shows
			elegant solution
			experimental results
			. . .
			what interests me here
			what interests us here
			...
				concluding
				studies to evaluate
				I have shown
				I showed that
				my goal was to
				. . .
				we report on our work on

h) Frase Auto-indicativa *(self-indicating phrases)*

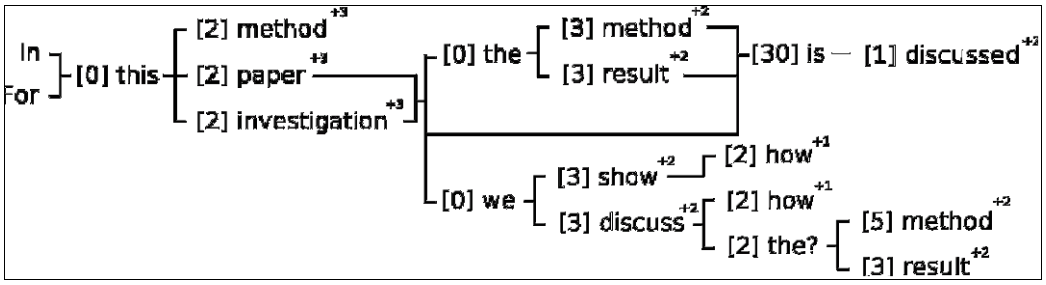
Alguns documentos, especialmente os discursivos, podem conter estruturas que ocorrem corriqueiramente e que, explicitamente, indicam que algo importante existe nas sentenças que a contém. As frases “The purpose of this article is...” e “The results of this article show that...”, exemplos de estruturas relevantes dentro do universo dos artigos científicos, são denominadas frases auto-indicativas.

Paice (1990) definiu um método de seleção destas frases auto-indicativas, com base na definição de sete estruturas básicas (*templates*). Estes *templates* foram identificados pelas letras A, B, C, D, E, F e G e compostos por uma sequência esperada de palavras, os “paradigmas”. A partir destas estruturas são atribuídos pesos à medida que sentenças do texto se enquadravam nestes modelos pré-definidos. A busca por segmentos no texto que se adequem a uma destas estruturas forma a base para o processo de identificação das frases auto-indicativas.

A identificação das frases é um processo de busca de padrões no texto. A Figura 12.6 exibe um modelo de estrutura para indicadores do grupo “B”. As palavras encontradas na figura representam os paradigmas, o símbolo “?” denota que este paradigma é opcional

e os pesos de cada um são mostrados em números sobrescritos (por exemplo, o valor -+2).

Figura 12.6. Estruturas para indicadores do grupo B - simplificado (Paice , 1990)



Cada palavra é representante de um grupo de palavras que podem aparecer em determinado ponto da estrutura. As estruturas são utilizadas com um conjunto de grupos de palavras, sendo os paradigmas os membros chave dos grupos. Exemplos de paradigma e respectivas palavras chave são exibidos na Tabela 12.3.

Tabela 12.3. Lista de paradigmas e grupos de palavra-chave

Paradigma	Grupo de Palavras-chave
Paper	Paper, article, report, review, analys-, discussion, summar-, outline, presentation, essay, stud-, survey.
Method	Method, means, model, technique, approach, process, program-, procedure, system, measure, formula, function, idea (-2)
Result	Result, outcome, finding, conclusion
Show	Show, shew, reveal, demonstrate, confirm, prove, indicate, explain, illustrate, illuminate, clarify, make clear, made clear, determine, discover, find, found, suggest (-2), propose (-2), argue (-2), see (-2).
How	How, why, what, whether, that.
The	Them, a, an, some, our (+2).

Dois principais problemas são apontados no cálculo deste atributo. O primeiro é que, apesar de ser bastante útil na maioria dos documentos, existe a possibilidade de não se encontrar nenhuma frase auto-indicativa em determinados textos. O segundo é que, com alguma frequência, frases que deveriam ser definidas como auto-indicativas podem conter em sua constituição palavras que não fazem parte da lista de indicadores, o que acaba por desconfigurar a localização de determinado padrão de estrutura pré-definido.

12.2.2.3. Redução de Dimensionalidade

O alto tamanho do vetor de palavras é um dos problemas mais significantes na classificação de textos. Por conta disso, antes que se inicie de fato o processo de classificação, deve-se tentar reduzir ao máximo o tamanho do vetor de termos sem, no entanto, influenciar negativamente na categorização.

Sebastianini (1999) distingue o processo de redução de dimensionalidade em dos diferentes tipos, por seleção de termos e por extração de termos, descritos nas seções subseqüentes.

a) Por seleção de termos

Técnicas para seleção de termos atentam para selecionar do conjunto original T, o sub-conjunto T', de termos que quando utilizados para classificação do documento apresente o melhor resultado possível.

Uma técnica simples é a da freqüência do documento de um termo, onde são mantidos apenas os termos que aparecem mais no documento, sendo removidos os que não influirão significativamente na classificação. Yang & Pedersen (1997) mostraram que é possível reduzir o tamanho em até dez vezes sem influência no resultado final.

b) Por extração de termos

Dado um determinado conjunto T' $\subset$ T, este método atenta para a geração de um conjunto de termos "artificiais" - T', do original T, que maximize o resultado da classificação. Uma das técnicas utilizadas é o agrupamento de termos (*Term Clustering*) que tenta agrupar as palavras que possuem alguma co-relação. Estes grupos de termos, ou um dos representantes dele, poderiam ser utilizados como dimensão dos vetores ao invés dos termos originais.

12.2.3. Mineração

Consideramos como mineração a etapa que congrega as atividades que atuam diretamente sobre o objetivo finalístico do processo. As aplicações práticas da mineração de texto são diversas, pode identificar automaticamente idiomas, classificar documentos em categorias ou agrupá-los por área de interesse (*clustering*), pode ser utilizada para identificar informação útil ou padrões em textos não estruturado, dentre uma série de outras utilizações.

Em função da complexidade que envolve cada uma destas áreas, neste trabalho, iremos abordar, como exemplo, apenas a técnica de classificação automática de textos. É importante ressaltar, entretanto, que algumas das atividades apresentadas podem ser utilizadas em outras áreas de aplicação da mineração de texto. Boa parte das etapas encontradas no processo de classificação poderia, por exemplo, ser partilhada com as atividades de agrupamento *(clustering).*

12.2.3.1. Classificação Automática de Textos

O processo de classificação de textos envolve a tentativa de, através de técnicas de aprendizagem de máquina, permitir aos computadores classificar novos documentos em categorias pré-definidas, a partir da inferência sobre um conjunto de documentos previamente classificados por humanos, chamado de grupo de treinamento.

Para Lacerda & Braga (2004) apud Cherkasskg & Mulier (1998), podemos definir um método de aprendizagem como:

> *Um algoritmo (usualmente implementado em software) que estima um mapeamento desconhecido (dependência) entre entradas e saídas dos sistemas baseado nos dados disponíveis, isto é, nas amostras (entradas, saídas) conhecidas. Uma vez que a*

dependência tem sido estimada, ela pode ser usada na predição das saídas futuras do sistema baseado nos valores de entrada conhecidos.

O núcleo do processo de classificação, portanto, é formado por um algoritmo base, baseado em aprendizado de máquina ou em regras estáticas, que define automaticamente a categoria do documento a ser classificado. Yang & Liu (1999) e Joachims (1998) realizaram um estudo comparativo de diversas técnicas de Aprendizado de Máquina no contexto da classificação de textos e, dentre as diversas técnicas apresentadas, duas se destacaram positivamente: a técnica do Vizinho mais próximo (*k-nearest neighbord*) e a de Máquinas de Vetores de Suporte (*Support Vector Machines*). Oliveira (2004) apresentou estudo sobre classificação com o algoritmo *Naive Bayes* com índices também positivos. Detalharemos estes três métodos, além dos métodos de Tabelas de Regras de Decisão e do algoritmo de Arvore de Decisão.

a) Vizinho mais próximo

O método do vizinho mais próximo vem sendo estudado no reconhecimento de padrões há algumas décadas. O raciocínio indutivo deste algoritmo corresponde a admitirmos que a classificação de uma instância "*x*" será muito similar à classificação de outra instância que se encontre próxima a esta, levando em consideração a distância euclidiana como conceito de proximidade.

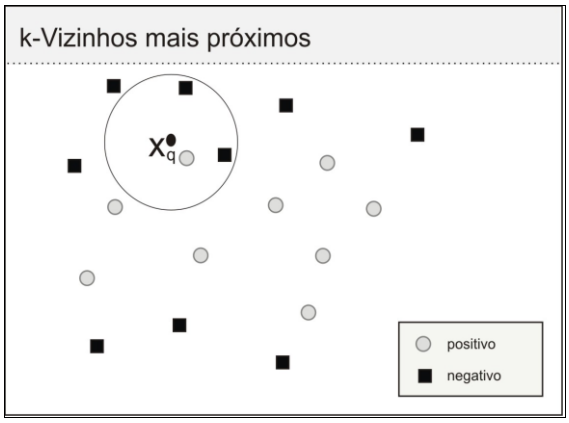
Tomemos como base a instância "*x*", supracitada, descrita pelo vetor de características $V = \{a_1(x), a_2(x), ..., a_n(x)\}$, onde $a_n(x)$ é o valor do enésimo atributo da instância "x". Desta forma, a distância entre duas instâncias x_i e x_j é definida no quadro 12.7.

Quadro 12.7. Distância entre as instâncias x_i e x_j.

$$d(xi, xj) \equiv \sqrt{\sum_{r=1}^{n}(a_r(x_i) - a_r(x_j))^2}$$

Observemos, por exemplo, o espaço bidimensional representado na Figura 12.7, que possui um ponto x_q a ser classificado como positivo ou negativo. Tomando como base os três vizinhos mais próximos, podemos claramente classificá-lo como negativo, já que a maioria dos vizinhos selecionados possui tal característica.

Figura 12.7. Método de classificação vizinho mais próximo.



O método do vizinho mais próximo tem se mostrado um método de inferência indutiva muito efetivo para diversos problemas práticos, sendo bastante robusto quando lhe é provido um conjunto de treinamento suficientemente grande.

Um dos problemas encontrados com este método, no entanto, é o fato de se utilizar todos os atributos da instância no momento da classificação. Em uma instância com 15 atributos onde apenas 3 são realmente relevantes, a distância entre os vizinhos levará em consideração todos os atributos, fazendo com que a classificação possa ser definida através dos 12 atributos que são irrelevantes, o que ficou conhecido como "maldição da dimensionalidade" (*curse of dimensionality*).

Soluções têm sido propostas como forma de resolver o problema da dimensionalidade. Dentre as encontradas, a utilização de técnicas para eliminação de atributos insignificantes e a utilização de pesos para diferenciar a importância dos termos tem trazido bons resultados.

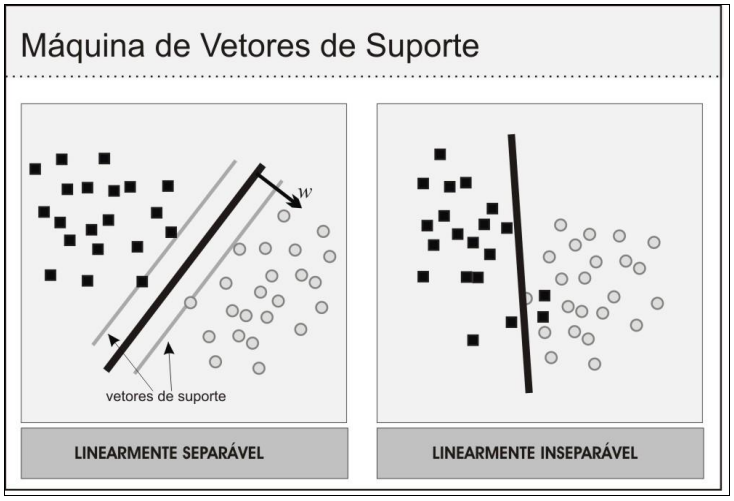
b) Máquinas de Vetores Suporte (Support Vector Machines - SVM)

O método em questão foi desenvolvido com base na teoria do aprendizado estatístico. O método das máquinas de vetores de suporte, doravante chamado SVM, é utilizado para o reconhecimento de padrões definidos sobre o espaço vetorial, onde o problema consiste em encontrar uma superfície de decisão que melhor separe os dados em duas categorias, a classe dos positivos e a dos negativos (MARTINS JÚNIOR, 2003).

Dado um conjunto de treinamento S, que contém pontos de duas classes, a meta é estabelecer a equação do hiperplano que divide S, deixando todos os pontos da mesma classe do mesmo lado, enquanto maximiza a distância mínima entre estas classes e o hiperplano.

Nem sempre essa separação dar-se-á de forma direta. Apenas nos casos linearmente separáveis será possível, a partir do conjunto de entrada, traçar um hiperplano que consiga dividir corretamente todos os integrantes de ambas as classes em seus respectivos subconjuntos. Nos casos linearmente inseparáveis, tanto o hiperplano quanto os vetores de suporte são obtidos através da resolução de um problema de otimização. Exemplos de ambos podem ser visualizados na figura 12.8.

Figura 12.8. Casos típicos encontrados com a aplicação do SVM.



$\vec{w} \cdot \vec{x} - b = 0, onde\ \vec{x}$ Considerando $\vec{w}$ o caso linearmente separável, podemos definir a superfície de $S_n = (\vec{x}_1, y_1), (\vec{x}_2, y_2) \ldots (\vec{x}_n, y_n)$ decisão do SVM como: é um ponto de dados arbitrário, que representa o $\vec{w}$ conteúdo do documento a ser classificado, e o vetor e a constante b devem ser aprendidos do conjunto de treinamento , de tamanho n e com y ε {-1, +1}. Como explanado, o problema do SVM é encontrar e b que satisfaçam as restrições encontradas no Quadro 12.8.

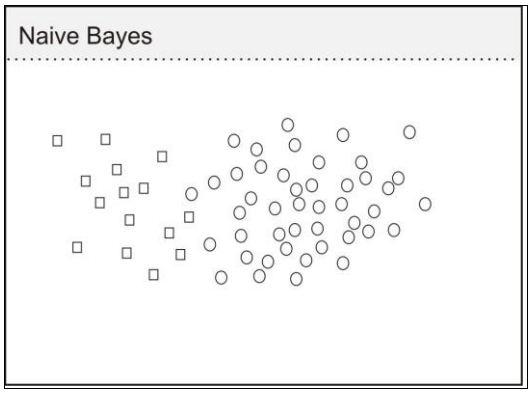
Quadro 12.8. Restrições do SVM

$\vec{w} \cdot \vec{x}_i - b \geq 1\ para\ y \in classe\ 1$ $\vec{w} \cdot \vec{x}_i - b \leq -1\ para\ y \in classe\ -1$

c) Naive Bayes

O teorema de Bayes, fundamentado na teoria da probabilidade, teve origem com o reverendo Thomas Bayes no século XVIII. De forma a facilitar o entendimento do teorema, discutiremos o exemplo descrito em Statsoft (2009). A figura 1.9. mostra um conjunto de objetos classificados em duas classes distintas, quadrado e círculo. O objetivo é descobrir a probabilidade que um novo objeto, aleatório, teria de pertencer a uma das classes dadas, baseado no conjunto de objetos existentes.

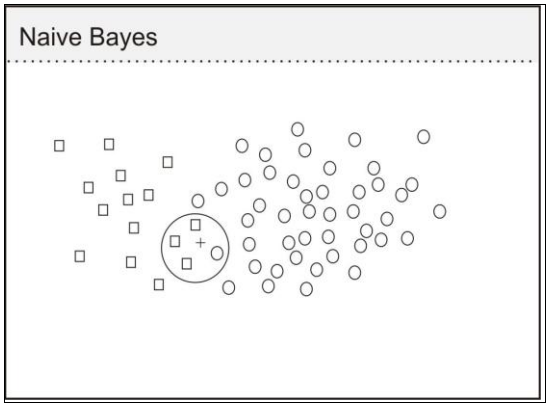
Figura 12.9. Grupo de Objetos divididos em duas classes



Uma vez que existem mais círculos do que quadrados, é correto acreditar que um novo caso possui mais chances de se encaixar entre os círculos. Na análise bayesiana, esta certeza é conhecida como probabilidade *a priori*, sendo baseada em experiências prévias, já que o conjunto de treinamento é conhecido. Pode-se, portanto, definir as probabilidades a priori como: *P(Círculo) = Número de círculos / número total de objetos e P(Quadrado) = Número de quadrados / número total de objetos*. Sabendo que existem 60 objetos, sendo 45 círculos e 15 quadrados: *P(Círculo) = 45/60 e P(Quadrado) = 15/60.*

Imagine, no entanto, que surja um novo ponto, uma cruz, entre os círculos e os quadrados, e que seja necessário determinar a probabilidade condicional dele pertencer a um dos grupos existentes, conforme apresentado na figura 12.10.

Figura 12.10. Classificação de novos dados utilizando Naive Bayes.



Supondo que os objetos estão agrupados, pode-se assumir que quanto mais objetos de determinada classe estejam próximos ao novo ponto, maiores as chances deste ponto estar associado a esta classe. Para medir esta probabilidade (do inglês *likelihood*), deve-se traçar uma circunferência ao redor da cruz, isolando internamente aqueles pontos que estejam mais próximos. A partir da contagem destes pontos vizinhos, por classe a que pertencem, pode-se obter a probabilidade que a cruz tem de ser círculo ou quadrado: *P(cruz, círculo) = Número de círculos vizinhos a cruz / total de círculos e P(cruz, quadrado) = número de quadrados vizinhos a cruz / total de quadrados*. Observando que existem dentro desta circunferência um círculo e três quadrados, tem-se: *P(cruz, círculo)= 1/45 e P(cruz, quadrado)=3/15.*

O teorema de Bayes trabalha, justamente, com a associação entre essas duas probabilidades. Na análise bayesiana, a probabilidade final vem da combinação de ambas as fontes de informação, formando a probabilidade *a posteriori*: *P(quadrado,cruz) = P(cruz, quadrado) * P(quadrado), P(quadrado,cruz) = 3/15 * 1/4 = 1/20, e P(círculo, cruz) = P(cruz, círculo) * P(círculo), P(círculo, cruz) = 1/45 * 3/4= 1/60.*

Desta forma, pode-se, finalmente, definir que a cruz deveria ser definida como quadrado, já que a probabilidade *a posteriori* do quadrado foi superior à do círculo.

O teorema de Bayes vem sendo utilizado como suporte ao processo de categorização de textos. No quadro 12.9, encontra-se a fórmula utilizada para estimar a probabilidade de um documento pertencer a uma classe (OLIVEIRA, 2004).

Quadro 12.9. Fórmula de Bayes aplicada à classificação de documentos

$$P(C = c_i \mid \vec{x}) = \frac{P(\vec{x} \mid C = c_i) \times P(C = c_i)}{P(\vec{x})}$$, **onde**:

$\vec{x}$ representa um vetor de termos e c_i representa uma classe..

d) Tabela de Regras de Decisão

Tanto as tabelas quanto as árvores de decisão não tem sua origem na área de sistemas de computação. Eles têm uma aplicação muito mais abrangente, englobando áreas como a Biologia e Teoria da Informação, além das Ciências da Computação.

Uma tabela de decisão é um modelo alternativo para um problema composto por condições, ações e regras. Condições são variáveis cujos valores são avaliados para a tomada de decisão. Ações representam o conjunto de operações a serem executadas condicionadas pelas respostas das condições. Regras representam o conjunto de situações, que são verificadas em resposta às condições (TCHEMRA, 2009).

Uma tabela de decisão é composta de linhas e colunas separadas em quatro quadrantes distintos, exemplificados no quadro 12.10. No quadrante superior esquerdo, cada linha corresponde à variável do problema que será analisada no processo de decisão – as condições; o superior direito corresponde aos valores das variáveis, das condições do primeiro quadrante; o inferior esquerdo indica o conjunto de procedimentos ou operações que serão executados de acordo com as respostas das condições; e o inferior direito contém as ações tomadas em razão da avaliação das condições.

Quadro 12.10. Quadrantes da tabela de decisão

Condições	Valores para as condições
Ações	Ações a serem tomadas

Para criar uma tabela de decisão é necessário definir as condições que podem afetar uma decisão e determinar as possíveis ações que podem ser tomadas. Em seguida, deve-se determinar o número de valores possíveis para cada condição para que seja preenchido quando cada ação for tomada.

Tomemos como exemplo uma mercearia que deseja criar uma tabela de decisão sobre as ações a serem tomadas quando for efetuada uma compra, dependendo do valor da compra e da forma de pagamento. As condições são:

(1) Compras abaixo de R$ 100

(2) Pagamentos com cheque

(3) Pagamentos com cartão de crédito

(4) Pagamentos em dinheiro

e as ações possíveis de serem tomadas para as condições são:

(1) Efetuar a venda

(2) Consultar o cheque

(3) Consultar o cartão de crédito

Usando os valores acima para construir uma tabela de decisão, deve-se definir os valores possíveis para cada condição e calcular as possibilidades. Para cada uma das condições, há apenas dois valores possíveis: Sim e Não. Como são quatro condições, com dois valores possíveis, chega-se a soma de oito possibilidades, visualizadas na Tabela 12.4

Tabela 12.4. Condições e Ações da tabela de decisão modelo (S/Atende e N /Não atende)

Compras acima de R$ 100,00	S	S	S	S	N	N	N	N
Pagamentos com cheque	S	N	N	S	S	N	N	S
Pagamentos com cartão de crédito	N	S	N	N	N	S	N	N
Pagamentos em dinheiro	N	N	S	S	N	N	S	S
Efetuar a venda			X		X	X	X	
Consultar o cheque	X			X				X

Consultar o Cartão de Crédito		X						

A partir da análise dos valores atribuídos às condições, são definidas as ações a serem tomadas.

e) Árvore de Decisão

Uma árvore de decisão é uma representação de uma tabela de decisão sob a forma de uma árvore. Tem a mesma utilidade da tabela, constituindo-se como uma maneira alternativa de expressar as mesmas regras que são obtidas quando se constrói a tabela (ÁRVORE DE DECISÃO, 2009).

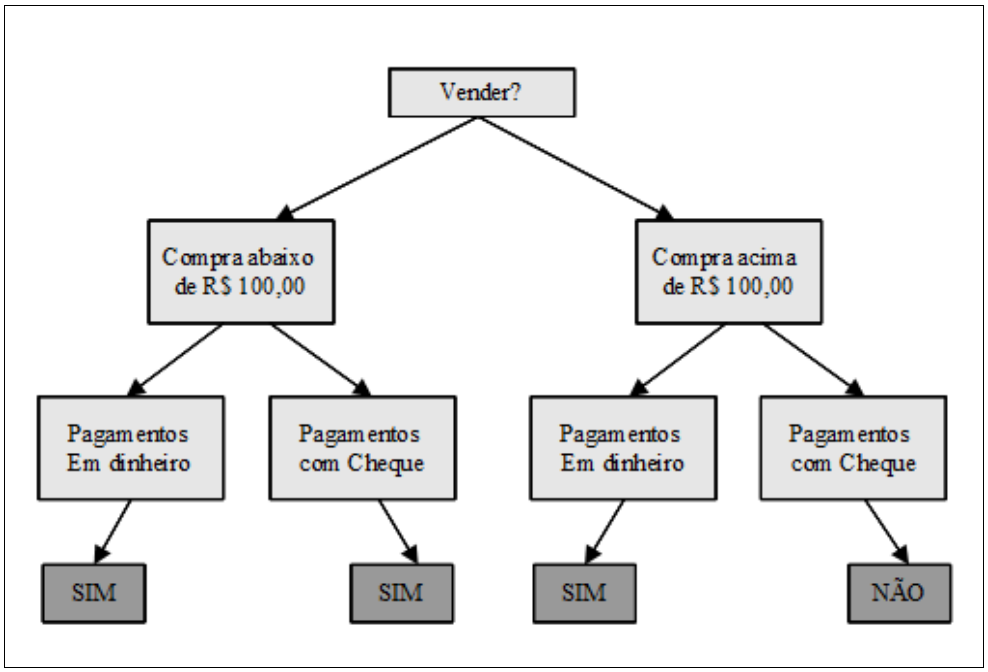
Em sua representação gráfica, a decisão (sim ou não) é representada por linhas e as questões sobre as quais se deve decidir são representadas por nós. Cada um dos ramos compostos por linhas e nós sempre termina em uma folha, indicando a consequência mais provável da sequência de decisões tomadas. Como exemplo, imagine que devemos decidir se uma venda pode ou não ser concretizada, dependendo do valor da compra e da forma de pagamento. As condições e ações a serem tomadas estão listadas na Tabela 1.5.

Tabela 12.5. Condições e Ações da tabela de decisão modelo

Valor Compra	Pagamento Dinheiro	Pagamento Cartão	Vende
Maior que R$ 100,00	S	N	S
Maior que R$ 100,00	N	S	N
Menor que R$ 100,00	S	N	S
Menor que R$ 100,00	N	S	S

A árvore de decisão correspondente pode ser visualizada na Figura 12.11.

Figura 12.11. Exemplo de Árvore de Decisão



As árvores de decisão, por sua representação gráfica, tornam mais fácil a visualização do problema e dos fluxos de decisão em comparação com a tabela de decisão. Este fato é válido quando consideramos um número pequeno de condições e ações; para conjuntos grandes, a árvore torna-se muito complexa e de difícil entendimento.

12.2.4. Apresentação

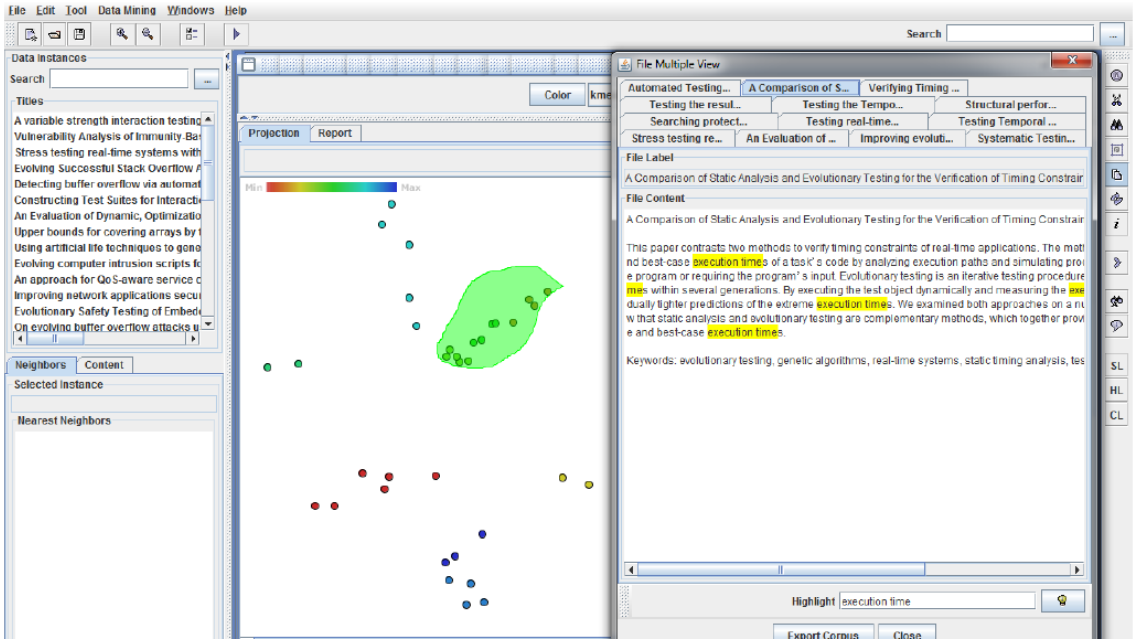
De uma forma geral, podemos visualizar a camada de apresentação como o meio de visualização das informações geradas pelos processos principais da mineração de texto, motivo pelo qual esta interface deve procurar ser o mais amigável possível, fazendo com que o resultado possa ser claramente manipulado pelos usuários. Provavelmente, a principal funcionalidade existente na camada de apresentação é a navegação pelos resultados da mineração. As ferramentas mais recentes possuem requisitos que possibilitam uma navegação dinâmica e direcionada ao conteúdo (Feldman; Sanger, 2006).

A utilização de ferramentas de visualização na mineração de mineração de texto, com o objetivo de facilitar e a navegação entre os conceitos e padrões, tem crescido vertiginosamente nos últimos pontos, a ponto de iniciar uma nova linha de pesquisa, denominada de Mineração Visual de Textos. Se antes os resultados eram apresentados em mapas e gráficos estáticos, as novas ferramentas de visualização permitem grande interatividade na manipulação, tanto dos dados de origem quanto das possibilidades de análise dos resultados.

A área de mineração visual de textos provê ferramentas gráficas que obtém vantagem das habilidades visuais dos usuários para dar suporte ao processo de aquisição do conhecimento. Como exemplo, podemos citar o trabalho de Felizardo et. al. (2010) , que propôs uma abordagem baseada na Mineração Visual de Textos para suporte a

categorização e classificação no mapeamento sistemático. Uma das telas da ferramenta desenvolvida – denominada Projection Explorer, pode ser visualizada na Figura 12.12.

Figura 12.12. Tela de resultados da ferramenta Projection Explorer



12.3. Estudo de Caso – Buscador Google

O Google, além de ser o principal serviço de buscas na internet, é o site mais acessado no mundo. A ideia dos fundadores do Google em desenvolver uma nova ferramenta de busca surgiu dos inúmeros problemas encontrados nos buscadores da época, que realizavam a pesquisa baseado apenas na análise das palavras chaves do texto, o que acarretava, com frequência, um resultado onde grande quantidade das páginas exibidas não tinha correlação com o objetivo do usuário na consulta. O principal diferencial desta nova ferramenta de busca em relação às demais era o fato de que, em complemento à utilização das palavras chave, a estrutura presente no hipertexto das páginas indexadas seria considerada no algoritmo de priorização do resultado da busca.

De um modo geral, podemos dividir o processo executado pela ferramenta de busca em três partes distintas e complementares: o de localização e captura dos documentos; o de indexação; e o de tratamento das consultas realizadas pelos usuários. Estas atividades serão descritas nas seções subseqüentes.

12.3.1. Localização e Captura dos Documentos

O primeiro passo se inicia com a captura dos documentos na web, que tem por objetivo encontrar e recuperar páginas que serão encaminhadas para indexação. Esta atividade é executada pelo Googlebot[12], de duas formas distintas, a primeira, percorrendo os hiperlinks

[12] Googlebot é o robô de busca do Google, um software responsável por coletar os documentos na web que

entre as páginas, e a segunda, capturando uma URL encaminhada através de cadastro no endereço http://www.google.com/addurl.html . Quando o Googlebot acessa uma página, ele captura todos os links existentes e adiciona a uma fila de espera para capturas subsequentes.

Apesar de se apresentar como um algoritmo simples, a ferramenta tem alguns desafios a enfrentar, como o controle de concorrência – já que chega a acessar milhares de páginas simultaneamente, e os agendamentos para reindexação de páginas, pois que o conteúdo disponibilizado pode mudar a qualquer momento. Páginas de jornais, por exemplo, devem ser reindexadas com uma frequência muito maior do que as páginas encontradas na maioria dos outros tipos de sites. O Google declara que, de uma forma geral, consegue, periodicamente, reindexar toda a sua base em três ou quatro meses[13].

12.3.2. Indexação dos Documentos

O processo de indexação dos documentos do Google guarda similaridades com o processo descrito neste capítulo, desde a utilização de uma série de operações de pré-processamento até a construção de um modelo intermediário, com conteúdo estruturado, onde serão executadas as operações de busca e priorização dos resultados.

O principal diferencial do Google em relação aos demais buscadores é o seu algoritmo de classificação e priorização dos resultados da busca, o qual chamou de cálculo do "pagerank" da página. Hoje, são consideradas cerca de duzentas diferentes regras para o cálculo do pagerank, mas a base de todo o algoritmo surgiu com a ideia da utilização não apenas das palavras chave na indexação, mas da estrutura presente no hipertexto para melhorar os resultados da busca.

Os autores esperavam que a utilização da estrutura de hipertexto pudesse melhorar o resultado da consulta, em particular, acreditavam que a estrutura e os textos dos links encontrados nas páginas proviam informação suficiente para realizar julgamentos relevantes sobre a qualificação das páginas e, por conseguinte, melhorar a qualidade da busca. A ideia é que quanto maior a quantidade de links, de referências para uma determinada página for encontrada em outras páginas, provavelmente, maior será a sua relevância. Esta ideia é estendida pelo pagerank não apenas contando os links de todas as páginas de forma igualitária, mas normalizando o resultado a partir da quantidade de links na página. A fórmula inicial do cálculo do pagerank pode ser visualizada no Quadro 12.11, sendo que *PR(X)* é o pagerank de *X*, *d* é um fator de ajuste e *C(X)* é a quantidade de hiperlinks existentes em *X (*BRIN; PAGE, 1998).

Quadro 12.11. Fórmula para cálculo do pagerank

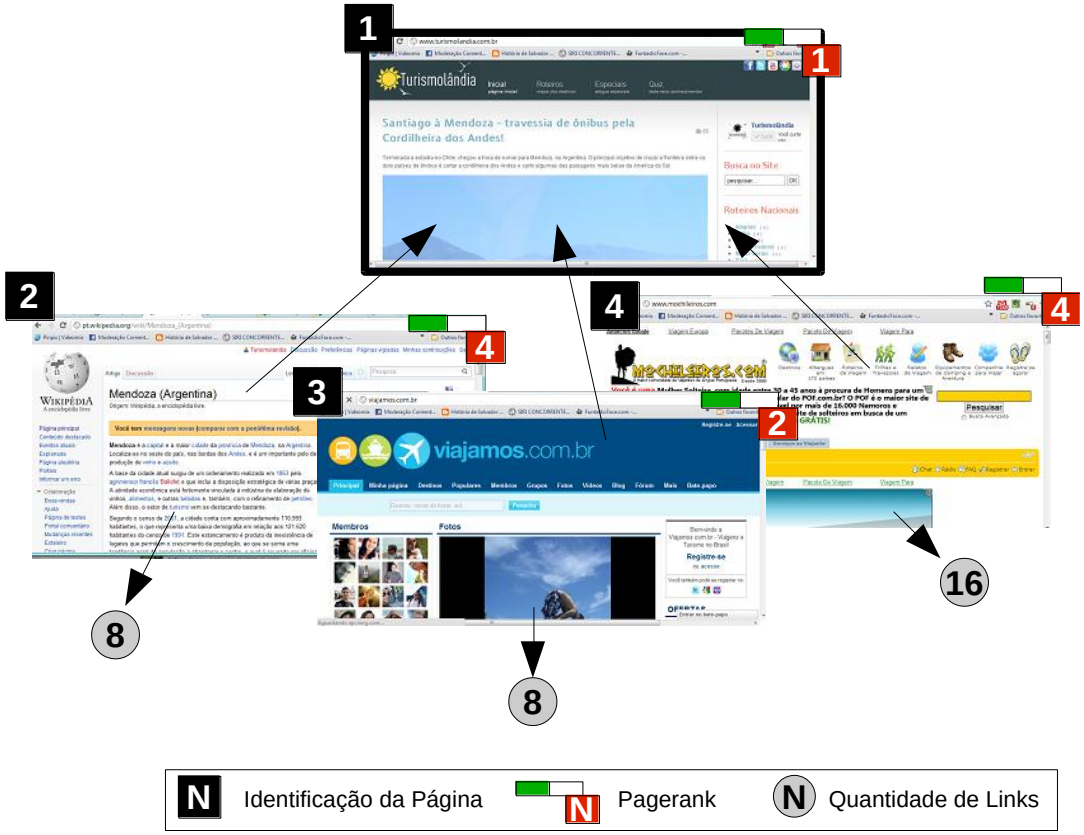
$$PR(A) = (1-d) + d\ (PR(T1)/C(T1) + \ldots + PR(Tn)/C(Tn))$$

serão indexados pela ferramenta de busca.

[13] http://www.google.com/about/company/tech.html acessado em 09/03/2012 às 10:00

A Figura 12.13. apresenta um exemplo do cálculo do pagerank. As páginas numeradas como 2, 3 e 4, encontradas pela ferramenta de busca em sua base de dados, são responsáveis por fornecer a base para o cálculo do grau de relevância da página descrita como 1. Com base nestas informações, o cálculo do Pagerank de 1 teria a seguinte composição: PR(1) = (1-0,85) + 0,85 (4/8 + 2/8 + 4/16) , por conseguinte, o PR(1) = 1.

Figura 12.13. Exemplo do cálculo do pagerank



O pagerank pode ser visto como um padrão de comportamento do usuário, a expectativa de uma pessoa imaginária, que está "surfando na internet" aleatoriamente, visitar uma página e continuar clicando em hiperlinks sem voltar a página anterior. O fator de ajuste "*d*" é a probabilidade, em um determinado momento, desta pessoa se cansar e requerer uma nova página inicial, randômica. O valor de "d" é definido entre 0 e 1, sendo comumente utilizado o valor 0,85.

Outra característica do algoritmo é que ele utiliza o texto presente no hiperlink na indexação da página que está sendo referenciada. De uma forma geral, estes textos se mostram como bons descritores da página de destino, além de prover informação útil para direcionamentos que não são passíveis de indexação, como imagens e programas, o que permite que a busca retorne resultados de endereços que não passaram pelo processo de indexação

12.3.2.1. Sistemas de Controle

Como parte do método do cálculo do pagerank das páginas é de conhecimento público, o Google enfrentou uma série de ações de usuários tentando burlar o algoritmo e fazer com que os seus sites obtivessem melhor colocação nos resultados da busca. Desta forma, foi obrigado a desenvolver algumas ferramentas de controle para evitar estas possíveis fraudes.

A empresa monitora sites que avançaram no ranking muito rapidamente e endereços que estão hospedados em servidores cujo endereço IP esteja na lista negra da companhia. Outra preocupação da companhia é com a "venda" de hiperlinks por páginas que tem um alto pagerank, para combater este tipo de fraude vem desenvolvendo estratégias para controlar o crescimento de hiperlinks que direcionam para endereços não relacionados ao tópico da página de origem, através do tratamento de palavras chave e da classificação do conteúdo da página.

Outras estratégias utilizadas por fraudadores são a de inundar a páginas com palavras irrelevantes, usar redirecionamentos, criar subdomínios e domínios com conteúdo similar ou efetuar spam de comentários em blogs com a url do seu site, este último já combatido com a criação da tag html "*nofollow*".

Estes são apenas alguns exemplos das inúmeras ferramentas de controle necessárias para conter ações maliciosas de usuários, e mostram alguns dos problemas encontrados quando lidamos com ferramentas de processamento automático.

12.3.3. Processamento de Consultas

De uma forma geral, o Google exibe páginas em que foram encontradas "todas" as palavras-chave enviadas na consulta do usuário, à exceção das "stop words". A ferramenta permite, ainda, que operadores lógicos, comuns em expressões regulares, sejam utilizados na busca, artifício que auxilia no tratamento de ambiguidades ligadas ao contexto.

Imagine, por exemplo, que um usuário deseje buscar informações relacionadas a fruta "manga". Utilizando na busca apenas a palavra-chave "manga", nenhum dos resultados iniciais retornados é relacionado ao objetivo inicial, todos os sites listados na primeira página estavam relacionados ao termo "Mangá", as conhecidas revistas em quadrinhos japonesas (Figura 12.14).

Com o objetivo de fornecer algum contexto ao processo de busca, informamos explicitamente na consulta que, para o resultado, devem ser descartadas as páginas em que a palavra manga está associada a palavra "anime". Desta forma, foi possível eliminar os resultados ligados ao termo "Mangá" (Figura 12.15).

Figura 12.14. Resultado da busca da palavra "manga" no Google

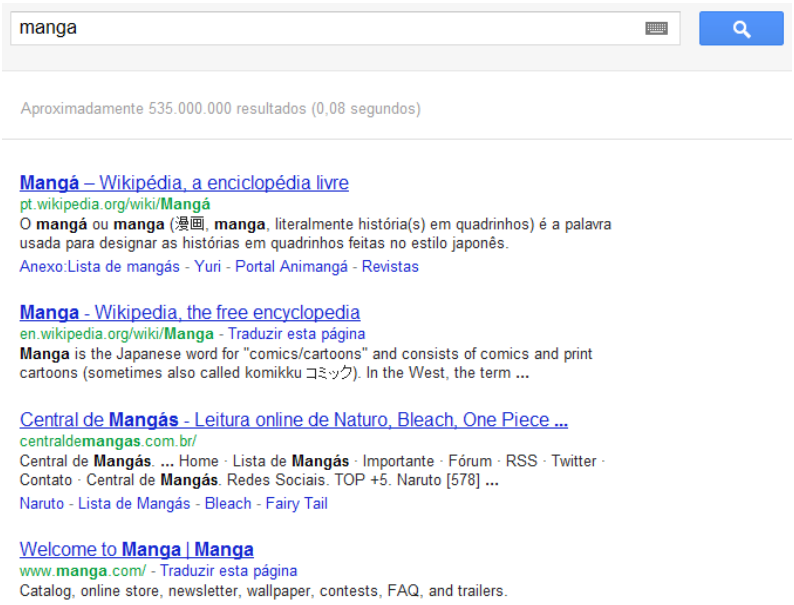
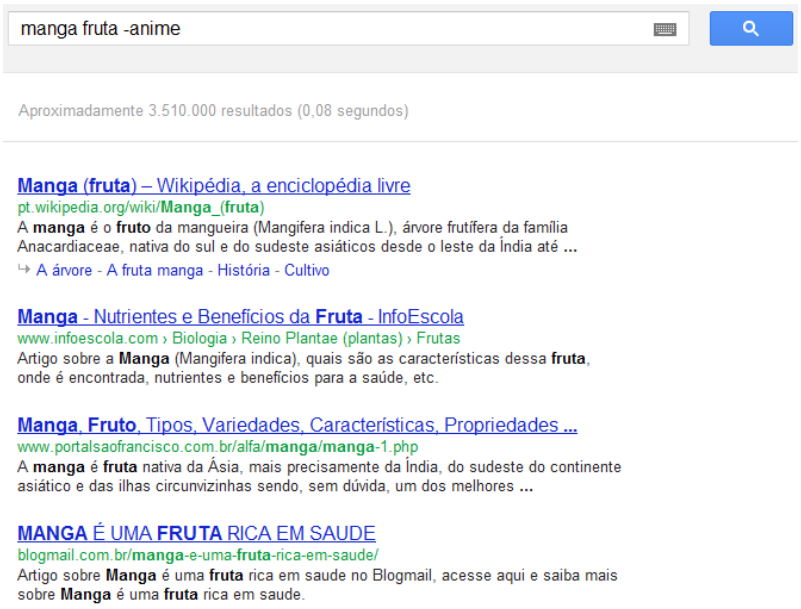


Figura 12.15. Resultado da busca da palavra “manga”, descartada a palavra anime, no Google

Com o resultado, observou-se que, mesmo eliminando com sucesso os resultados ligados ao Mangá, não foi possível atingir o objetivo inicialmente desejado, já que o resultado não priorizou os sites onde a palavra manga se relacionasse com uma fruta. Incrementando um pouco a busca, a palavra “fruta” foi adicionada na consulta. O resultado pode ser visualizado na figura 12.16.

Figura 12.16. Resultado da busca da palavra “manga” e “fruta”, descartada a palavra anime, no Google



O resultado satisfatório demonstra que a forma de construção da consulta é diretamente responsável pela qualidade dos resultados, além de esclarecer parte do funcionamento do sistema de tratamento de consultas utilizado pelo Google. Por vezes pode-se pensar, quando obtemos um retorno da ferramenta fora do esperado, que o processo de indexação e priorização das páginas foi realizado de forma errônea, o que tem-se que entender, entretanto, é que o processo de mineração e indexação das páginas, de uma forma geral, não é capaz de tratar o contexto. Inovações implantadas pelo Google já avaliam o perfil de navegação do usuário de forma a priorizar a busca sem uma definição explícita do contexto, conforme pode ser visualizado em sua nova política de privacidade[14]. Uma proposta pública de processo para construção de perfis de usuário a partir de dados de navegação pode ser encontrada em Torres (2005).

De forma a aperfeiçoar o resultado da busca, o Google também utiliza técnicas de "*stemming*" e descarte de palavras comuns (*stop words*) no tratamento das consultas. A consulta da expressão "*swimming sea*", por exemplo, traz resultados para páginas que tem como palavras chave "*swim"* e "*sea"*. É relevante ressaltar, entretanto, que o algoritmo de cálculo de relevância (pagerank) provavelmente dará maior importância aos sites que contém a palavra exata.

Já em relação às *stop words*, em função de não representarem significado relevante no contexto desta ferramenta de busca, não são tratadas como palavras chave na consulta, a menos que o usuário deseje buscar uma expressão exata, ação que é executada com a utilização de aspas duplas. Observe que os resultados para consulta das expressões "viajar de salvador" (Figura 12.17) e "viajar para salvador" (Figura 12.18) trazem resultados com páginas que tratam de um mesmo assunto, mesmo que com uma ordenação diferente, para dados de entrada que possuem diferentes significados.

[14] http://www.google.com.br/intl/pt-BR/policies/privacy/ acessado em 10/03/2012 às 11:20

Figura 12.17. Resultado da consulta para a entrada “Viajar de Salvador”

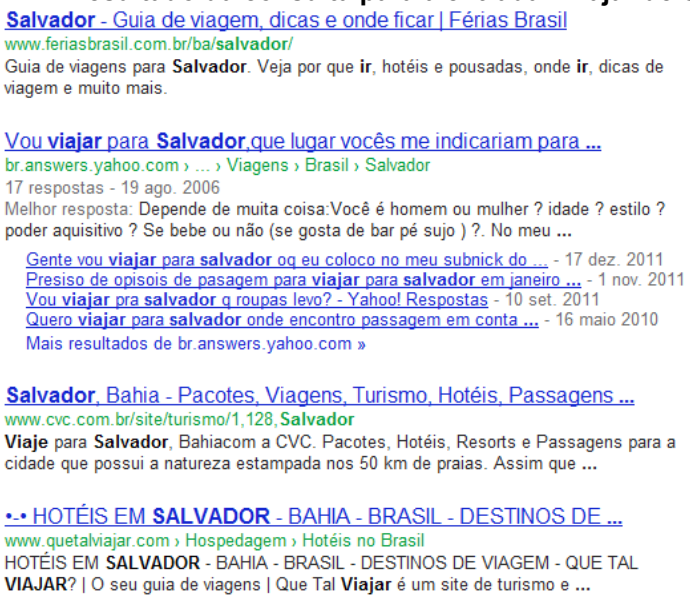


Figura 12.18. Resultado da consulta para a entrada “Viajar para Salvador”

A comprovação do descarte das *stop* words pode ser percebida pelo destaque que é dado as palavras-chave “viagem” e “Salvador”, que aparecem em negrito, conquanto as palavras “de” e “para” não são ressaltadas.

Durante a pesquisa, o Google avalia, ainda, a distância entre as palavras-chave nos documentos indexados e a ordem em que aparecem, de modo que páginas que possuam as palavras na mesma ordem da consulta irão ser priorizadas em relação as demais. Para executar esta operação, durante o processo de indexação, a informação sobre a seqüência das palavras dos textos analisados deve ser preservada.

12.4. Conclusão

O principal objetivo deste trabalho foi apresentar o universo da Mineração de Textos, sua arquitetura e principais processos de uma forma dinâmica e clara. Foram discutidas as principais atividades relacionadas a cada uma das etapas do processo – pré-processamento, mineração e apresentação, na tentativa facilitar o entendimento com a utilização de exemplos práticos da aplicação.

O conteúdo apresentado não tem a pretensão de esgotar o tema, que é vasto, mas de servir para iniciação aos estudos na área de Mineração de Textos. Deve ser utilizado como um breve guia para contextualizar o domínio de conhecimento sobre o tópico. Uma análise pormenorizada e mais específica sobre o funcionamento das atividades, entretanto, requer a pesquisa e utilização de documentos adicionais. Neste contexto, a seção de referências deste trabalho se apresenta como uma boa alternativa para guiar estudos complementares.

12.5. Referências

ÁRVORE DE DECISÃO. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2011. Disponível em: <http://pt.wikipedia.org/w/index.php?title=%C3%81rvore_de_decis%C3%A3o&oldid=26382926>. Acesso em: 4 set. 2009.

BAXENDALE, P. B. Machine-made index for technical literature - An experiment. IBM Journal of Research and Development, 2(4), 354–361, 1958.

BRIN, S., PAGE, L. Anatomy of a large-scale hypertextual web search engine. In Proceedings of the 7th International World Wide Web Conference (Brisbane, Australia, Apr. 14 –18). pp. 107–117. 1998

CROFT, W. B. Clustering large files of documents using single-link method. Journal of the American Society for Information Science, 28:341-344, 1977.

EDMUNDSON, H. P. New methods in automatic extracting. Journal of the Association for Computing Machinery, 16(2) 264-285, April 1969

FELDMAN, R., SANGER, J. The text mining handbook. Cambridge University Press, 2006.

FELIZARDO, K. R.; NAKAGAWA, E. Y. ; FEITOSA, D. ; MINGHIM, R. ; MALDONADO, José Carlos . An Approach Based on Visual Text Mining to Support Categorization and Classification in the Systematic Mapping. In: 13th International Conference on Evaluation & Assessment in Software Engineering (EASE 2010), 2010, Staffordshire

FRAKES, W. e BAEZA-YATES, R. Information retrieval, data structures and algorithms. Prentice Hall, New York, Englewood Cliffs, N.J., 1992.

FRANCIS, W. e KUCERA, H., 'Brown Corpus Manual'. Disponível em http://khnt.aksis.uib.no/icame/manuals/brown/ . Acesso em: 01 mar. 2012.

HUI, B. Developing and Evaluating a Document Visualization System for Information Management. Master Thesis, University of Toronto, January 2002.

HULL, D. A. and Grefenstette, G. A detailed analysis of english stemming algorithms. Technical Report TR MLTT-023. RXRC, Rank XEROX, 1996.

INDERJEET, M.; GATES, B.; BLOEDORN, E. Improving summaries by revising them. In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99), pages 558–565, 1999.

KROVETZ, R. Viewing Morphology as an Inference Process, In Proceedings of ACM-SIGIR93, pp191-203, 1993 .

JOACHIMS, T. Text categorization with support vector machines: Learning with many relevants features. In Proc. 10th European Conference on Machine Learning (ECML'98), pages 137--142, Chemnitz, Germany, 1998.

KOSINOV, S. "Evaluation of N-grams conflation approach in text-based information retrieval". String Processing and Information Retrieval, 2001. SPIRE 2001. Proceedings.Eighth International Symposium on Volume , Issue , 13-15 Nov. 2001 Page(s): 136 – 142.

KUPIEC, J.; Pedersen, J.; Chen, F. A trainable document summarizer. Proc. SIGIR, ACM Press, 68–73, 1995.

LACERDA, W. S.; e BRAGA, A. P. Experimento de um Classificador de Padrões baseado na Regra Naive de Bayes. Anais da VI SECICOM, Lavras - Brasil, p.30-35, 2004

LAROCCA NETO, J.; SANTOS, A.D.; KAESTNER, A.A., FREITAS, A. A. Generating Text Summaries through the Relative Importance of Topics. In: Proceedings of the International Joint Conference IBERAMIA/SBIA, Atibaia, SP, 2000.

LIN, C. Training a selection function for extraction. Proceedings of the 18th Annual International ACM Conference on Information and Knowledge Management. pp. 55–62, 1999.

LUHN, H. P. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development, 2, 157-165, 1958.

MANNING, C. D. Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In Alexander Gelbukh (ed.), Computational Linguistics and Intelligent Text Processing, 12th International Conference, CICLing 2011, Proceedings, Part I. Lecture Notes in Computer Science 6608, pp. 171--189. Springer, 2011.

MARCHI, A. R. Projeto lacio-web: Desafios na construção de um corpus de 1,1 milhão de palavras de textos jornalísticos em português do brasil. Em 51o Seminário do Grupo de Estudos Lingüísticos do Estado de São Paulo, São Paulo, Brasil, 2003.

MARCUS, M.; SANTORINI, B.; e MARCINKIEWICZ, Maryann. "Building a large annotated corpus of English: the Penn Treebank." Computational Linguistics, 19(2), 1993

MARTINS, C.B.; Pardo, T.A.S.; Espina, A.P.; Rino, L.H.M. Introdução à Sumarização Automática. Relatório Técnico RT-DC 002/2001. Departamento de Computação, Universidade Federal de São Carlos. São Carlos-SP, Fevereiro, 38p, 2001

MARTINS JUNIOR, J. Classificação de páginas na internet. 89 f. Tese (Mestre em Ciências da Computação e Matemática Computacional), Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo, São Carlos, 2003.

OLIVEIRA, G. L. Categorização Automática de Documentos por Múltiplos Classificadores Naive Bayes. 157 f. Tese (Mestre em Redes de Computadores, área de concentração Gestão do Conhecimento), Universidade Salvador, Salvador, 2004.

PAICE, C. D. Another stemmer, SIGIR Forum, 24(3), 56-61, 1990.

PARDO, T.A.S. Gistsumm: um sumarizador automático baseado na idéia principal de textos. Série de Relatórios do Núcleo Interinstitucional de Linguística Computacional, São Paulo, 2002.

PARDO, T.A.S; Rino, L.H.M.; Nunes, M.G.V. NeuralSumm: Uma Abordagem Conexionista para a Sumarização Automática de Textos. In Anais do IV Encontro Nacional de Inteligência Artificial – ENIA, pp. 1-10. Campinas-SP, Brasil, 2003

PORTER, M.F. An algorithm for suffix stripping. Program, 14(3), 130-137, 1980.

SEBASTIANI, F. Machine Learning in Automated Text Categorization. Technical Report IEI B4-31-12-99, Instituto di Elaborazione della Informazione, Consiglio Nazionale delle Ricerche, Pisa, Itália, 1999.

STATSOFT. Naive Bayes Classifier – Introduction Overview. Disponível em http://www.statsoft.com/textbook/stnaiveb.html, acesso em capturado em 14/08/2009.

STEIN, B; e POTTHAST, M. Putting Successor Variety Stemming to Work. In Reinhold Decker and Hans J. Lenz, editors, Advances in Data Analysis, pages 367-374, 2007. Springer. ISBN 978-3-540-70980-0.

TCHEMRA, A. H. Tabela de decisão adaptativa na tomada de decisões multicritério. 172 p. Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais, São Paulo, 2009.

TEUFEL, S. e Moens, M. Sentence Extraction as a Classification Task, Workshop 'Intelligent and scalable Text summarization, ACL/EACL 1997, July 1997.

TORRES, J.A.S. Proposta de Arquitetura para Construção de Perfis de Usuário através da Mineração da Web. Monografia (Graduação em Análise de Sistemas) - Universidade do Estado da Bahia, Salvador, 2005.

YANG Y. e LIU X. A re-examination of text categorization methods. In Proc. 22th ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR'99), pages 42-49, Berkeley, CA, 1999.

YANG, Y.; e PEDERSEN J.P. A Comparative Study on Feature Selection in Text Categorization. Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), pp412-420, 1997.

Capítulo 13

Conhecendo a Computação Móvel Sensível ao Contexto

Marcio Pereira de Sá

Abstract

Context-Aware Computing has became a reality and this just was possible due to the popularization of mobile devices such as tablets and smartphones. Mobile applications can be more useful if they use information about their execution environment such as date and time, location and power level of their mobile devices. These applications are named context-aware applications and take into consideration their execution environment and adapt their behavior according to current context.

Resumo

A Computação Sensível ao Contexto tem se tornado uma realidade e isto só foi possível devido à popularização dos dispositivos móveis, como tablets e smartphones. Aplicações móveis podem ser mais úteis se estas usarem informações sobre seu ambiente de execução, como a data e a hora, a localização do dispositivo/usuário e o nível de energia dos dispositivos móveis. Essas aplicações são denominadas aplicações sensíveis ao contexto e levam em consideração seu ambiente de execução e adaptam seu comportamento de acordo com o contexto atual.

13.1. Introdução

A Computação Sensível ao Contexto tem por objetivo permitir o desenvolvimento de aplicações que percebam e reajam a mudanças do ambiente no qual estão inseridas como, por exemplo, alterações na localização de um usuário, na temperatura de uma determinada sala e no estado dos recursos computacionais de um dispositivo. Todas essas informações são denominadas informações de contexto e são providas, direta ou indiretamente, por sensores.

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

Além disso, pode-se dizer também que a Computação Sensível ao Contexto se caracteriza pelas pesquisas e o desenvolvimento de aplicações que levam em conta todo o ambiente de execução em que estão envolvidas. Toda informação sobre o ambiente que as envolve usada pela aplicação para melhorar a interação com seus usuários.

Por outro lado, Computação Móvel é a área da Computação responsável pelas pesquisas relacionadas ao desenvolvimento tanto de dispositivos móveis (*hardware*), tais como aparelhos telefônicos celulares, *smartphones*, *tablets*, livros digitais e navegadores GPS, quanto de aplicações e sistemas que são executados nesses dispositivos (*software*). É atualmente uma das áreas da Tecnologia da Informação mais importantes, pois tem experimentado, nas últimas duas décadas, um crescimento espetacular a ponto de alguns considerarem a Computação Móvel como a *Quarta Revolução da Computação* [Loureiro 1998], antecedida pelos grandes centros de processamento de dados da década de 1960, o surgimento dos terminais nos anos 1970 e as redes de computadores e a popularização dos computadores pessoais na década de 1980.

Uma outra subárea da Computação também muito importante atualmente é Computação Ubíqua e Pervasiva. A Computação Ubíqua, através da definição dada por Mark Weiser [Weiser 1991], se refere ao uso coletivo de computadores disponíveis no ambiente físico dos usuários, talvez embutidos em uma forma invisível para esses usuários. É a computação pervasiva, onde os sistemas computacionais estão à disposição de seus usuários praticamente em qualquer lugar e a qualquer momento.

É a união dessas duas importantes áreas da Computação (Computação Móvel e Computação Ubíqua) que permite o desenvolvimento de programas de computador ou aplicações computacionais mais úteis e acessíveis a seus usuários. Dentre essas aplicações móveis e ubíquas mais conhecidas, estão os serviços baseados em localização (*LBS - Location-based Service*). Estas aplicações fornecem informações a seus usuários de acordo com o local em que se encontram. Como exemplo, poderíamos utilizar um sistema que notificasse seus usuários sempre que estes estivessem próximos a lojas, pontos turísticos ou de lazer de interesse desses usuários. Estão também entre as categorias de aplicações móveis mais populares e interessantes os jogos eletrônicos para *smartphones*, as agendas de compromissos e as redes sociais para dispositivos móveis.

Atualmente, há vários tipos de dispositivos móveis disponíveis no mercado, tais como aparelhos telefônicos celulares comuns, telefones inteligentes (*smartphones*), PDAs (*Personal Digital Assistant - Assistente Digital Pessoal*), *tablets*, rastreadores com receptores GPS e leitores de livros eletrônicos (*e-readers*). Apesar da diversidade de dispositivos móveis existentes, um grande nicho, em especial, se abre para os profissionais da Computação Móvel: o mercado de telefones inteligentes ou *smartphones*. Esses aparelhos possuem sistemas operacionais próprios e recursos de hardware (telas sensíveis ao toque, acelerômetros, receptores GPS, câmeras fotográficas, dentre outros) que viabilizam o desenvolvimento de aplicações móveis cada vez mais adaptadas ao ambiente e a seus usuários.

Por isso, através da união com a Computação Móvel, uma nova e promissora área da Computação está surgindo, a Computação Móvel Sensível ao Contexto. Esta visa proporcionar aos usuários de dispositivos móveis serviços mais úteis e "inteligentes", capazes de se tornarem o principal diferencial das novas aplicações móveis que estão sendo criadas.

13.2 Computação Móvel

Como dito na Seção 1.1, a Computação Móvel se ocupa com as pesquisas e o desenvolvimento tanto de dispositivos móveis (*hardware*), quanto de aplicações e sistemas que são executados nesses dispositivos (*software*). Nesse sentido, é importante analisar aspectos tecnológicos, tanto de hardware quanto de software, quando se aprofunda no estudo desta área da Computação. Em relação ao hardware, algumas características são especialmente importantes para o nosso estudo, como as diferentes gerações tecnológicas da telefonia móvel. Aspectos importantes de software incluem as aplicações móveis e suas plataformas de desenvolvimento, como o iOS (iPhone e iPad) e o Android.

Estes e outros aspectos são tratados nas subseções seguintes.

13.2.1. Breve Histórico da Computação Móvel

De acordo com [Loureiro 1998], em linhas gerais, a Computação Móvel tem como precursores os sistemas analógicos de comunicação que têm, em sua origem remota, a descoberta feita por Hans Christian Oersted, em 1820, de que a corrente elétrica produz um campo elétrico. A partir daí, foi possível desenvolver vários sistemas de comunicação, cujo primeiro foi o *telégrafo*. Este sistema, inicialmente baseado na comunicação com fio, ainda na metade do século XIX, permitia a transferência de palavras a longa distância através do código Morse. Em seguida, após a descoberta das equações de Maxwell, que descrevem a propagação de ondas eletromagnéticas, aliada aos experimentos de Heinrich Hertz, foi possível a descoberta da radiotelegrafia por Marconi, no final do século XIX. Em 1901, o Oceano Atlântico era atravessado por sinais de rádio. Estavam lançadas as bases para a comunicação sem fio.

A segunda geração de sistemas de comunicação desenvolvida foi o *telefone*, inventado por Alexander Graham Bell. Em seguida, com o advento dos computadores, temos a terceira geração dos sistemas de comunicação, pois, através dos sistemas computacionais, a comutação telefônica também se torna digital, reduzindo de forma drástica a necessidade de operadores humanos no sistema de comutação telefônico. Até este momento, ainda predominava a comunicação com fio, caracterizada pelo elevado custo de acesso remoto. Por causa disso, os sistemas sem fio se tornaram atraentes. Porém, nesta época, ainda dependiam significativamente das redes fixas.

De modo geral, o primeiro sistema de comunicação móvel existente foi um sistema de rádio utilizado pela polícia de Detroit, nos Estados Unidos, em 1928. Já em 1947, a AT&T desenvolve o IMTS (*Improved Mobile Telephone Service*), um sistema de transmissão em que era necessário somente uma única torre de alta potência para atender a uma grande área ou cidade. Posteriormente, nos Anos 1970, a AT&T lança o sistema celular denominado AMPS (*Advanced Mobile Phone System*). No início, era um sistema utilizado em automóveis e atendiam um pequeno número de usuários com uma capacidade muito limitada de tráfego. O Japão foi o país que recebeu a primeira rede celular no

mundo, no final da década de 1970, mais precisamente em 1979. A segunda geração do sistema AMPS aparece em 1983, com a primeira rede celular americana que operava em Chicago e Baltimore.

A década de 1990 foi relativamente produtiva para a telefonia celular. Já em 1991, acontece a validação inicial dos padrões TDMA e CDMA, nos Estados Unidos e a introdução da tecnologia microcelular. Um ano mais tarde, em 1992, aparece o sistema Pan-Europeu GSM (*Groupe Spéciale Mobile*). Em meados dessa década, temos a introdução do sistema CDPD (*Cellular Digital Packet Data*) e início dos serviços PCS (*Personal Communication Services*) CDMA e TDMA, em 1994. Os projetos para a cobertura terrestre por satélites de baixa órbita, como o projeto Iridium são iniciados em 1995. A década de 2000 é marcada principalmente pela popularização do 3G e das redes Wi-Fi.

13.2.2. As Gerações da Telefonia Móvel

De maneira geral, podemos organizar a história e a evolução da telefonia móvel em gerações. Estas gerações são caracterizadas principalmente por suas tecnologias de comunicação. A seguir, são listadas as gerações e suas características.

- *Primeira Geração - 1G*

Esta foi a Primeira Geração de telefonia móvel. Dentre suas características principais, destacam-se o emprego de tecnologia analógica e o uso de aparelhos volumosos em relação aos aparelhos celulares atuais. Os principais padrões empregados pela primeira geração foram: AMPS (*Advanced Mobile Phone System*), desenvolvido nos Estados Unidos, ainda na década de 1970. É considerado o primeiro padrão de rede celular. Um outro padrão semelhante é o TAC (*Total Access Communication system*), uma versão europeia do padrão AMPS. Temos ainda o ETACS (*Extended Total Access Communication System*), considerado uma evolução do TAC.

- *Segunda Geração - 2G*

A Segunda Geração de telefonia móvel tem como característica principal a passagem da tecnologia analógica para a digital. Consequentemente, houve também uma diminuição do tamanho e do peso dos aparelhos celulares. Dentre os padrões de segunda geração, destacam-se: GSM (*Global System for Mobile communications*), era considerado o principal padrão 2G na Europa e utiliza as bandas de frequência de 900 MHz e 1800 MHz (na Europa) e 1900 MHz (nos Estados Unidos). O CDMA (*Code Division Multiple Access*) e o TDMA (*Time Division Multiple Access*) são tecnologias alternativas ao GSM europeu.

Outra característia importante das tecnologias 2G é a capacidade de transmitir além da voz, dados digitais de pouco volume, como mensagens de texto curtas (SMS - *Short Message Service*) ou mensagens multimídia (MMS - *Multimedia Message Service*). O padrão GSM original permite transferência a, no máximo, 9,6 Kbps.

- *Gerações 2.5G e 2.75G*

O GPRS (*General Packet Radio System*) foi desenvolvido como uma extensão do padrão GSM, permitindo, teoricamente, velocidades de transferência de dados até 114

Kbps. Por ser uma evolução do padrão GSM de segunda geração, o GPRS recebeu a denominação de 2.5G. Já o padrão EDGE (*Enhanced Data Rates for Global Evolution*) conseguiu elevar a velocidade de transferência teórica para cerca de 384 Kbps, tornando viável o uso de algumas aplicações multimídias. Por isso, o EDGE ficou conhecido como um padrão 2.75G.

- *Terceira Geração - 3G*

Dentre as principais características da Terceira Geração de telefonia móvel (3G), especificadas pelo IMT-2000 (*International Mobile Telecommunications for the year 2000*) da União Internacional das Comunicações (UIT), destacam-se: altas taxas de transmissão de dados, compatibilidade mundial e compatibilidade dos serviços de terceira geração com os sistemas de segunda geração. O principal padrão 3G é denominado UMTS (*Universal Mobile Telecommunications System*); este padrão utiliza uma banda de frequência de 5 MHz, podendo alcançar velocidades de transferência de dados de 384 Kbps a 2 Mbps. O padrão UMTS emprega a codificação W-CDMA (*Wideband Code Division Multiple Access*).

Uma evolução do padrão 3G é a tecnlogia HSDPA (*High-Speed Downlink Packet Access*), considerado um padrão 3.5G. O HSDPA permite taxas de transferência entre 8 a 10 Mbps.

- *Quarta Geração - 4G*

A Quarta Geração da telefonia móvel está baseada em dois padrões principais: o LTE (*Long-Term Evolution*) e o Wi-Max. As grandes vantagens dos serviços e sistemas de quarta geração são a compatibilidade e padronização das tecnologias em nível mundial, bem como um grande aumento nas velocidades de conexão em relação aos padrões atuais (3G e 3.5G). A expectativa é de que se tenha velocidades de transmissão de até 100 Mbps. Porém, apesar de já ser uma realidade em outros países, no Brasil, a implantação efetiva de redes 4G não deverá ocorrer, pelo menos, nos próximos dois anos.

13.3. Computação Ubíqua

Imagine que, ao entrar em uma sala de reuniões, na empresa onde trabalha, seu telefone celular automaticamente mudasse o tipo de toque de campainha para “silencioso”, pois o seu aparelho “inteligente” sabia que você (através de sua agenda de compromissos, hospedada na Web), a partir daquele momento até às dez e trinta da manhã, estaria em uma reunião muito importante e não desejaria que fosse incomodado de modo abrupto, evitando assim contrangimentos. Imagine ainda que, no final da tarde, ao chegar em casa, o sistema de climatização de sua residência automaticamente acionasse os condicionadores de ar, deixando a residência na sua temperatura preferida. Logo após, a campainha tocasse com um rapaz entregando-lhe algumas compras feitas também de forma automática por sua geladeira que notou a carência desses produtos.

Este tipo de comportamento, até certo ponto futurista, já não existe apenas na imaginação visionária de alguns cineastas, nem é privilégio de alguns pesquisadores, trabalhando em complexos laboratórios de pesquisa de novas tecnologias. Atualmente,

com o avanço da computação móvel e das técnicas de computação ubíqua, isto já é uma realidade!

Porém, desenvolver sistemas computacionais que possam se comportar de modo mais "inteligente" e mais útil aos seus usuários só é possível se tais sistemas forem dotados de capacidades parecidas com aquelas encontradas nos seres vivos, especialmente os seres humanos, como a capacidade de sentir o ambiente a sua volta e reagir às mudanças neste ambiente. Sistemas computacionais dotados dessas características são, então, denominados *Sistemas ou Aplicações Sensíveis ao Contexto*, pois usam as mudanças nas informações de contexto, como localização, data e hora, agenda de compromissos, dentre outras para reagir e adaptar seu comportamento de acordo com tais mudanças.

Entretanto, deve-se observar que não apenas os grandes e complexos sistemas computacionais, como casas inteligentes e sistemas de segurança de algumas instituições financeiras, podem se tornar sensíveis ao contexto. O uso de informações de contexto para melhorar a interação com usuários já é empregado há muito tempo em aplicações relativamente simples, como em sistemas de busca pela Web, aplicações para o envio de anúncios inteligentes, sistemas de tradução de textos, etc.

Da perspectiva da Computação Móvel, o uso da sensibilidade ao contexto oferece muitas oportunidades para a criação de aplicações mais adaptadas ao ambiente móvel e ubíquo, porém há também muitos desafios para desenvolvê-las. Dentre estes desafios estão a grande diversidade de informações contextuais (localização dos usuários, luminosidade ambiente, uso de CPU e memória, qualidade da rede sem fio e muitas outras) e a abundância de tecnologias de sensoriamento, como, por exemplo, vários modelos de sensores de temperatura, cada um com uma interface de acesso própria ou ainda diferentes APIs para acesso a agendas de compromissos de usuários, hospedadas na Web. Nota-se também que, em geral, as informações de contexto necessárias para desenvolver estas aplicações podem ser obtidas de muitas formas distintas, como através do uso de sensores físicos para a obtenção de dados sobre a temperatura ou luminosidade do ambiente, informações sobre o status do dispositivo (quantidade de memória livre, nível de uso da CPU, qualidade da rede); ou através ainda de sensores lógicos, fornecendo informações sobre a agenda de compromissos dos usuários, suas preferências, suas páginas Web mais visitadas, dentre outras.

Por isso, desenvolver aplicações sensíveis ao contexto sem nenhuma infraestrutura de provisão de contexto não é tarefa simples, exigindo que o programador dedique grande parte de seu tempo em tarefas relacionadas à coleta, processamento e disponibilização de informações contextuais. Esse tempo precioso dedicado a essas tarefas poderia ser melhor empregado em atividades mais diretamente relacionadas à lógica da própria aplicação a ser desenvolvida em vez de gastá-lo em tarefas relacionadas à provisão de informações de contexto. Um outro problema desse estilo de programação é a dificuldade em manter a aplicação, pois se um sensor tiver que ser substituído por outro com uma interface de acesso aos dados diferente, o programador deverá modificar boa parte do código de sua

aplicação para que a mesma possa agora se comunicar com o novo sensor e receber dele os dados coletados.

Portanto, para facilitar o desenvolvimento de aplicações sensíveis ao contexto e popularizar o uso dessas aplicações junto aos usuários finais, é necessário o uso de infraestruturas para a provisão de contexto (plataformas de *middleware*, *frameworks*, *toolkits*, etc.). Essas infraestruturas fornecem mecanismos que facilitam a coleta, processamento e disponibilização de informações de contexto às aplicações, permitindo que os programadores possam se concentrar, principalmente, em assuntos mais diretamente relacionados à lógica específica de sua aplicação, deixando assuntos inerentes à provisão de contexto, como a comunicação adequada com todos os sensores, o processamento dessas informações e a sua disponibilização sob a responsabilidade dessas plataformas de provisão de contexto.

13.3.1. Classificação, Histórico e Evolução da Computação Sensível ao Contexto

De acordo com o exposto anteriormente, os sistemas computacionais que utilizam informações contextuais para reagir e se adaptar a mudanças no ambiente que os circunda são denominados sistemas ou aplicações sensíveis ao contexto e, por conseguinte, a subárea da Computação que estuda e desenvolve tais sistemas é denominada *Computação Sensível ao Contexto*. Deve-se ressaltar, ainda, que a Computação Sensível ao Contexto é, na verdade, uma subárea dentro de uma área mais ampla denominada *Computação Ubíqua ou Pervasiva* que, por sua vez, está situada na subárea da *Computação Geral* denominada *Redes de Computadores e Sistemas Distribuídos*. Além disso, a união da Computação Sensível ao Contexto com a Computação Móvel dá origem a uma nova e promissora área, conhecida por *Computação Móvel Sensível ao Contexto*.

A Figura 13.1 ilustra essa classificação.

Para Mark Weiser [Weiser 1991], a Computação Ubíqua está relacionada ao uso coletivo de computadores disponíveis no ambiente físico dos usuários, podendo até estar embutido em uma forma invisível para estes mesmos usuários. Uma outra definição semelhante é a da Computação Pervasiva que se refere à visão de dispositivos ou computadores fazendo parte efetiva da vida das pessoas. Este fenômeno pode ser visto, por muitos, como uma combinação da Computação Móvel (ou seja, o uso de computadores instalados no próprio corpo dos usuários ou que podem ser carregados por estes) e os computadores embutidos nos ambientes fixos, podendo, por este fato, ser compreendida como sinônimo para a computação ubíqua [Loke 2006].

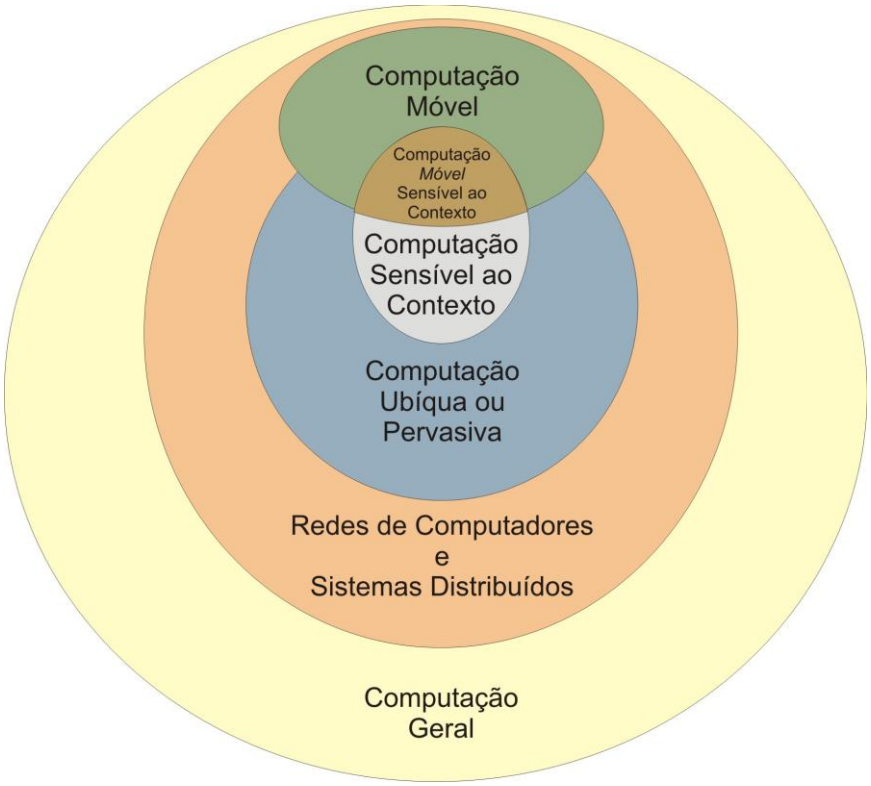


Figura 13.1. O relacionamento entre a Computação Sensível ao Contexto e outras áreas da Computação Geral.

Segundo Dey *et al* [Abowb 1999], o termo "*sensibilidade ao contexto*" foi usado, pela primeira vez, em 1994, no trabalho dos pesquisadores B. Schilit e M. Theimer [Schilit 1994]. Neste trabalho, os autores se referiam à *sensibilidade ao contexto* como sendo a capacidade que um software possui para se adaptar ao local em que está sendo executado, ao conjunto de pessoas e objetos próximos, e também às mudanças dessas pessoas e desses objetos ao longo do tempo. Porém, antes mesmo dessa publicação, ainda em 1992, um trabalho pioneiro já estava sendo desenvolvido: o *Active Badge Location System*, da Olivetti [Want 1992] (um sistema baseado em localização). Este e outros trabalhos contemporâneos são, de alguma forma, derivados da visão visionária de Mark Weiser [Weiser 1991], ainda em 1991, sobre uma nova forma de desenvolver e utilizar software, com capacidades avançadas de interação tanto com os usuários como com o ambiente em que é executado.

Para ilustrar essa pequena história da Computação Sensível ao Contexto e, por conseguinte, da Computação Ubíqua, uma linha do tempo sobre alguns fatos importantes dessas duas áreas é exibida na Figura 13.2.

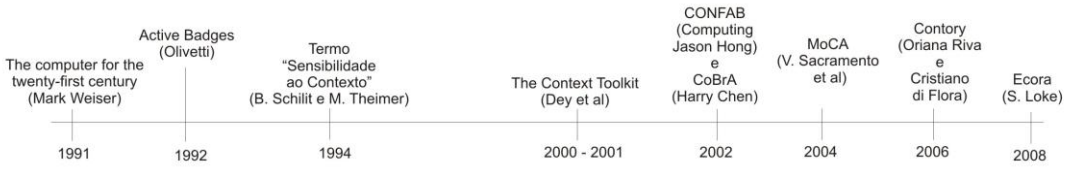


Figura 13.2. Alguns fatos importantes sobre a evolução da Computação Sensível ao Contexto.

13.3.2. O que é Contexto

Conforme observado por [Loke 2006], a noção de contexto tem sido empregada em inúmeras áreas, incluindo linguística, filosofia, representação do conhecimento, teoria da comunicação e resolução de problemas no campo da inteligência artificial. Portanto, é natural encontrarmos várias definições de contexto, de acordo com a área de interesse. De modo geral, por exemplo, o *Free Online Dictionary of Computing* [Dictionary 2011] define contexto como "aquilo que está em volta, e dá significado a algo." Com certeza esta é uma definição bastante ampla e não nos auxilia muito quando o assunto é especificamente a área de Sistemas Distribuídos e Redes de Computadores.

Nota-se, como mencionado anteriormente, que o trabalho de M. Theimer e B. Schilit, referia-se a contexto como sendo a informação de localização, a identidade das pessoas e objetos colocalizados e ainda as mudanças nesses objetos. Entretanto, ao longo dos anos seguintes, vários pesquisadores têm procurado definir o significado do termo "Contexto" sob o ponto de vista da Computação. Ryan *et al*. [Ryan 1997] definem, de modo similar, o termo contexto como sendo a localização do usuário, o ambiente, a identidade e o tempo.

Dey *et al*. [Abowb 1999], nos dá uma definição mais operacional de contexto, que nos parece ser mais útil na prática e mais adequada à computação ubíqua. Para eles, *contexto é qualquer informação que possa ser usada para caracterizar a situação de entidades (por exemplo, pessoas, locais ou objetos) que são consideradas relevantes para a interação entre um usuário e uma aplicação, incluindo também o usuário e a aplicação*. No restante desse capítulo, esta será a definição que empregaremos ao usar este termo.

A definição de Dey *et al*., descrita acima, é ilustrada na Figura 13.3.

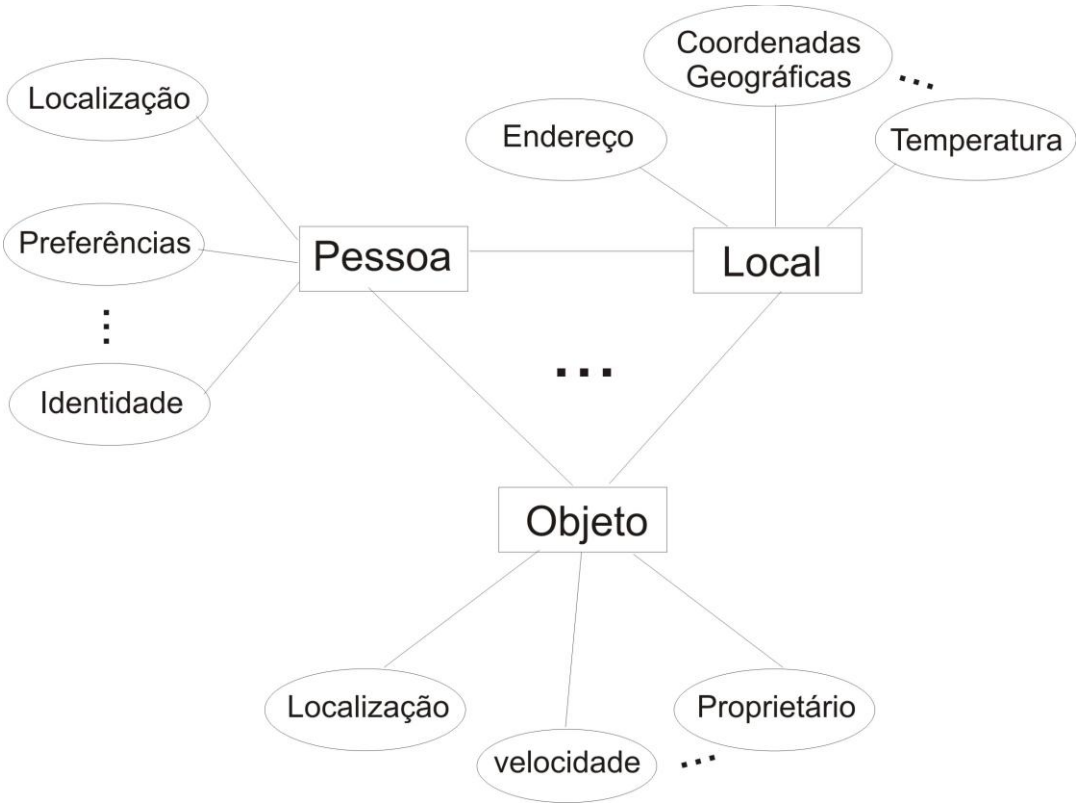


Figura 13.3. A definição de contexto segundo Dey *et al*.

13.3.3. Tipos e Categorias de Contexto

De modo semelhante à definição de contexto, durante anos, vários pesquisadores procuraram classificar as informações de contexto segundo alguns critérios e aspectos. A seguir, analisamos algumas dessas classificações.

T. Gu *et al*. [Gu 2005] admitem dois tipos ou categorias principais de contexto: *contexto direto* e *contexto indireto*. Para eles, *contexto direto* se refere àquele obtido ou adquirido diretamente de um provedor de contexto que pode ser uma fonte interna, como um provedor de localização interno (*indoor*) ou uma fonte externa, como um servidor de informações sobre meteorologia. Além disso, subdividem o contexto direto em contexto que pode ser sentido (*sensed context*) e contexto definido (*defined context*). O *sensed context* é adquirido ou obtido por sensores físicos, tais como a medida da temperatura de uma sala obtida por um sensor posicionado no teto da mesma, ou por sensores virtuais, como um web service fornecendo informações sobre a temperatura em uma determinada cidade distante. O *defined context*, por outro lado, é tipicamente "definido" pelo próprio usuário, como seus restaurantes ou produtos preferidos.

O contexto indireto é derivado através da interpretação de contexto direto (*context reasoning*). Por exemplo, o estado atual de uma pessoa (tomando banho) pode ser inferido de sua localização (está no banheiro), o estado do chuveiro (ligado) e o estado da porta do banheiro (fechada).

A Figura 13.4 exibe os diferentes tipos ou categorias de contexto discutidas por T. Gu *et al* [Gu 2005].

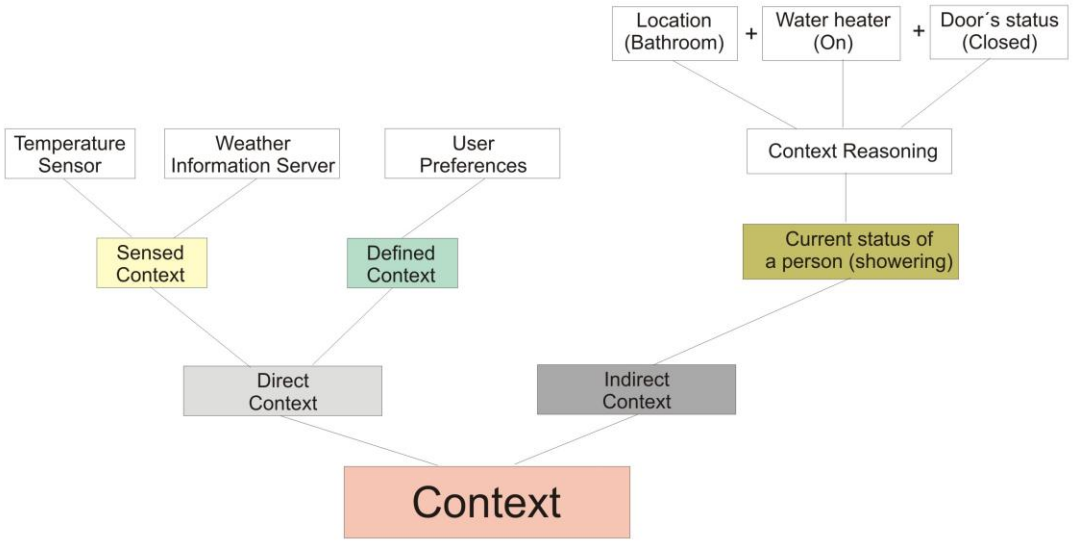


Figura 13.4. Os diferentes tipos ou categorias de contexto descritas por T. Gu *et al.*

De modo diferente, Schilit *et al*. [Bill 1994] descreve a existência de três categorias de contexto, a saber:

Contexto Computacional, ou seja, uso de CPU e memória, largura de banda, custos de comunicação, conectividade da rede e outros.

Contexto de Usuário que engloba questões como localização, perfil do usuário, situação social atual.

Contexto Físico, isto é, luminosidade, níveis de ruído, condições de tráfego e temperatura, dentre outros.

A Figura 13.5 descreve essas três categorias.

Guanling Chen e David Kotz [Chen 2000] ainda propõem uma outra categoria, o *contexto temporal*, tal como a hora do dia, semana, mês e, até mesmo, a estação do ano.

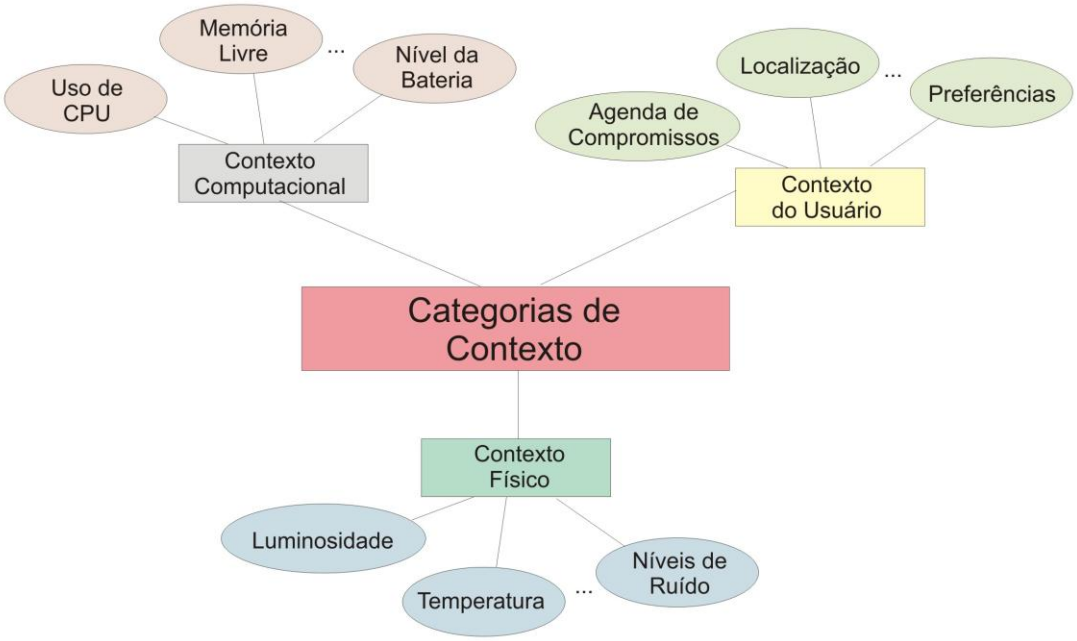


Figura 13.5. As três categorias de contexto descritas por Schilit *et al.*

13.3.4. Sistemas Sensíveis ao Contexto

Em geral, os organismos vivos são dotados de mecanismos de sensibilidade que permitem coletar informações sobre o ambiente que os envolve. Insetos, plantas e, especialmente, o ser humano dependem dos orgãos de sentidos (tato, olfato, paladar, visão e audição) para perceber o meio em que vivem e, em seguida, reagir da forma mais adequada a esse ambiente, que pode ser hostil, agradável, perigoso, etc. Esta capacidade de perceber o mundo e reagir a suas mudanças é vital para a sobrevivência dos seres vivos e, sem dúvida, pode ser vital para a sobrevivência das aplicações computacionais do Século XXI.

Por isso, é cada vez maior a importância dada à construção de sistemas computacionais verdadeiramente sensíveis ao contexto. Para Seng Loke [Loke 2006], um Sistema Pervasivo Sensível ao Contexto deve ter três funcionalidades básicas: *Sensibilidade*, *Processamento* e *Execução de ações*. De acordo com ele, os sistemas podem até variar o grau de sofisticação em cada uma dessas funcionalidades, mas todas elas devem sempre existir. A Figura 1.6 ilustra esta divisão.

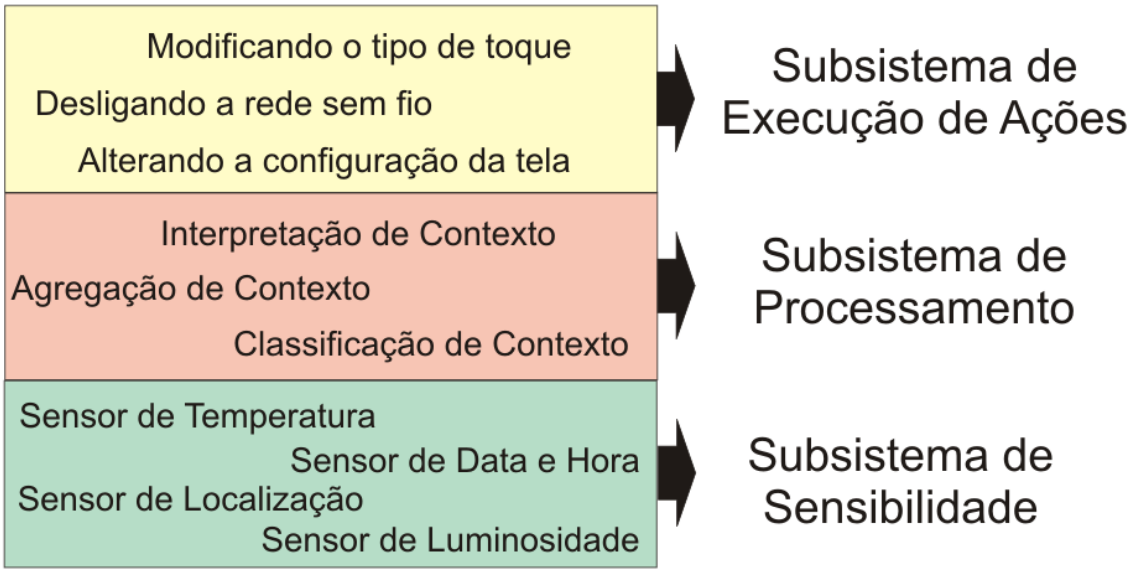


Figura 1.6. A estrutura interna básica de um sistema sensível ao contexto.

13.3.4.1 Sensibilidade (*Sensing*)

De fato, os sensores são os mecanismos responsáveis por coletar ou adquirir dados ou informações sobre o mundo físico ou algum aspecto deste. Existem sensores biológicos e não biológicos, os quais podem ser combinados para tornar a compreensão do mundo físico ou de parte dele o mais coerente possível com a realidade.

Além disso, é possível e, até, desejável em muitos casos, o uso de múltiplos sensores a fim de fornecer uma maior quantidade de informações relevantes às aplicações sensíveis ao contexto, permitindo processamento e conclusões mais complexas e mais reais. Seng Loke [Loke 2006] considera os sensores como uma ponte entre o mundo físico e o mundo virtual dos programas de computador. Para ele, esta é a grande diferença entre um software convencional e um sistema sensível ao contexto, visto que em um software convencional, a entrada de dados normalmente deve vir manualmente através dos usuários. Porém, em um sistema sensível ao contexto, os sensores são os maiores responsáveis pela entrada de informações, automatizando ao extremo a execução de aplicações.

Deve-se observar também que tipo de informação pode ser coletada por um sensor. Nos últimos anos, com o desenvolvimento da tecnologia da informação, microeletrônica e outras, várias espécies de sensores estão sendo desenvolvidas, como sensores de movimento, localização, temperatura e luminosidade. Além desses, uma grande quantidade de dispositivos podem ser considerados sensores, como microfones, medidores de pressão e relógios internos de computadores (*computer clock*).

De forma simplificada, pode-se dividir os sensores em dois grandes grupos: os *sensores físicos* e os *sensores lógicos*. Os sensores físicos são os dispositivos físicos (*hardware*) capazes de coletar dados de alguma variável física do ambiente, como localização, luminosidade, umidade, pressão atmosférica, temperatura. Por outro lado, os sensores lógicos, são, na verdade, aplicações computacionais (*software*) capazes de coletar alguma informação que, em geral, não pode ser coletada por dispositivos físicos, como a

agenda de compromissos de um usuário, suas preferências, suas páginas Web mais visitadas, dentre outras.

Devido à grande diversidade de informações de contexto, há uma enorme quantidade de sensores existentes e um dos desafios ou preocupações dos desenvolvedores de aplicações sensíveis ao contexto em ambientes ubíquos é a localização desses sensores.

Os sensores podem ser instalados no próprio ambiente físico (como sensores de luminosidade, temperatura, nível de ruído, etc.), em automóveis, nos dispositivos móveis, como PDAs e *smartphones*, que acompanham os usuários e, até mesmo, serem instalados em roupas, na pele ou dentro dos próprios usuários, como sensores médicos cardíacos, pílulas inteligentes.

Finalmente, é importante destacar que, quase sempre, é útil esconder como o tipo de informação de contexto foi adquirida, pois isso torna as aplicações que usam tais informações menos sensíveis às mudanças na forma como essas informações de contexto são coletadas.

13.3.4.2 Processamento (*Thinking*)

Após os dados contextuais terem sido coletados pelos sensores, eles podem ser imediatamente utilizados pelas aplicações ou serem processados para que novas formas de conhecimento possam ser extraídas desses dados brutos e serem utilizadas de forma mais eficiente pelas aplicações. De fato, os dados e informações capturados pelos sensores podem não ser suficientes para auxiliar as aplicações e os usuários em suas atividades.

Muitas vezes, é preciso agregar informações de contexto mais elementares ou até mesmo inferir informações de contexto mais complexas a partir de dados mais simples para que estas possam, de fato, ser úteis às aplicações. De modo geral, há três tipos principais de processamento de dados coletados pelos sensores: *interpretação*, *agregação* e *classificação*.

A *interpretação de contexto* tem a função de inferir novas informações contextuais a partir de outras já existentes. Isto é, obtém informações contextuais de mais alto nível a partir de dados de contexto de mais baixo nível. Por exemplo, um interpretador de contexto poderia, a partir das coordenadas geográficas fornecidas por um receptor GPS, inferir se um usuário específico se encontra ou não em sua residência ou em seu escritório num determinado momento.

Já a *agregação de Contexto* tem a responsabilidade de reunir informações contextuais inter-relacionadas, permitindo que estas informações possam ser manipuladas como uma única unidade. Como exemplo, pode-se citar um conjunto de informações de contexto relativas a uma determinada sala. Nela, há vários sensores coletando dados sobre temperatura, luminosidade, ruído e umidade. Por serem informações fornecidas por diferentes fontes contextuais, estes dados são tratados internamente como informações disjuntas. Um agregador de contexto poderia reunir todos esses dados em uma única

unidade de contexto relacionada à sala de onde essas informações estão sendo coletadas, facilitando assim o trabalho de interpretação e utilização desses dados.

A *classificação de contexto* é empregada para organizar os dados contextuais obtidos dos sensores em grupos e subgrupos de modo a facilitar tanto o trabalho dos agregadores quanto dos interpretadores de contexto. Pode-se, por exemplo, classificar as informações de contexto através das entidades (pessoas, objetos, locais, etc.) às quais tais informações então associadas, ou ainda classificar todas as informações de contexto relativas a um determinado usuário (localização, pessoas colocalizadas, preferências, dentre outras).

Não se deve confundir a classificação de contexto com a agregação de contexto. Classificar é realizar um processamento prévio nas informações, de modo a tornar mais fácil e rápido a identificação dessas informações, através de vários aspectos (por entidades, por precisão, etc.). Entretanto, agregar é reunir várias informações relacionadas a uma única entidade, por exemplo, para que posteriormente se possa manipulá-las como uma única informação composta.

Deve-se ressaltar, além disso, que muitas vezes, é necessário persistir os dados de contexto, de modo a oferecer dados históricos que possam ser úteis às aplicações posteriormente. Por exemplo, pode ser importante para uma determinada aplicação, ou mesmo para um interpretador de contexto, analisar as variações que uma determinada variável contextual sofreu durante um período específico, permitindo, a partir desses dados, fazer previsões sobre o que poderia acontecer ao ambiente em um intervalo de tempo subsequente.

13.3.4.3. Execução de Ações (*Acting*)

Uma vez que as informações de contexto são colhidas ou algumas situações são reconhecidas, ações devem ser executadas [Loke 2006]. É importante notar que as ações a serem executadas são específicas de cada aplicação. Muitas vezes, essas ações devem ser executadas imediatamente, antes mesmo que as situações que causaram a necessidade da execução dessas ações sejam alteradas.

Como exemplo, imagine uma aplicação baseada em localização utilizada para notificar usuários de dispositivos móveis sobre a presença de lojas que ofereçam determinados produtos de interesses para estes usuários. Se enquanto o usuário estiver se movimentando com seu automóvel, a aplicação descobrir alguma loja que ofereça o produto desejado pelo usuário, esta deverá avisá-lo imediatamente, pois, caso contrário, se houver demora nessa notificação, esta informação poderá já não ser tão útil caso o usuário já se encontre em uma outra região da cidade.

13.3.5. Desenvolvendo uma arquitetura para sistemas sensíveis ao contexto

Durante os últimos anos, várias propostas de arquiteturas para sistemas sensíveis ao contexto foram desenvolvidas. A Figura 13.7 mostra uma arquitetura abstrata em camadas proposta por Baldauf e Dustdar **[**Baldauf 2007**].** Para eles, o subsistema de sensibilidade é

composto pelos sensores propriamente ditos e por uma camada superior de recuperação de dados brutos, ou seja, ainda sem processamento algum. O subsistema de processamento é constituído pelas camadas de pré-processamento (em que os dados brutos são organizados) e o pelo processamento propriamente dito; além disso, há uma camada superior responsável pelo armazenamento e gerenciamento das informações processadas pela camada inferior.

Finalmente, as aplicações sensíveis ao contexto compõem o subsistema de execução de ações, onde, de fato, todas as ações ou reações ocasionadas pelas respectivas mudanças no ambiente que envolve as aplicações são executadas.

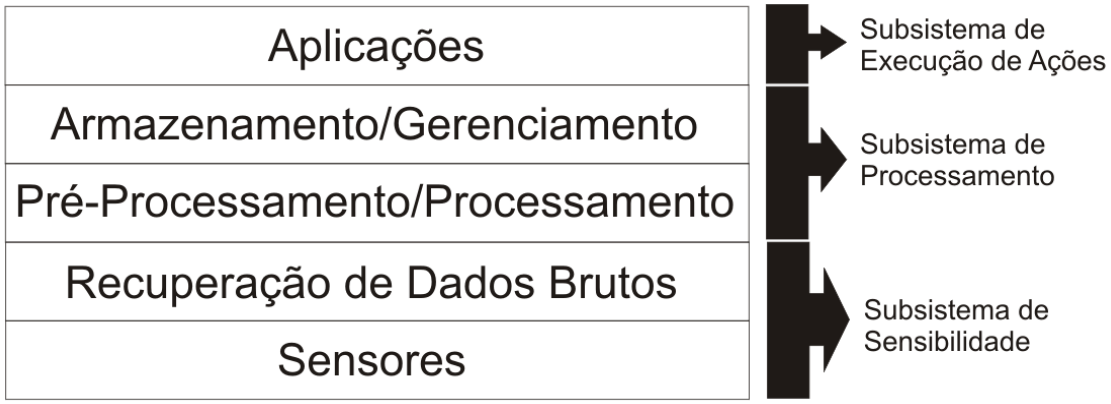


Figura 13.7. Uma arquitetura abstrata em camadas para sistemas sensíveis ao contexto.

13.4 Infraestruturas Para Provisão de Contexto

A provisão e o processamento de informações de contexto são tarefas difíceis e complexas. Esta afirmação se torna mais clara à medida que verificamos a diversidade de fontes de contexto (sensores) e a heterogeneidade das informações de contexto disponíveis ou necessárias, bem como dos ambientes de execução existentes.

Para solucionar ou, pelo menos amenizar, o problema anterior, há uma grande quantidade de pesquisas cuja finalidade principal é o desenvolvimento de camadas de software e hardware capazes de atuarem como intermediárias entre as aplicações e os sistemas computacionais para obtenção, tratamento e disponibilização das informações de contexto. Essas camadas intermediárias são conhecidas como infraestruturas para provisão de contexto. A seguir, são analisadas algumas das principais e mais significativas infraestruturas para provisão de contexto disponíveis.

13.4.1. Context Toolkit

O Context Toolkit [Salber 1999] [Dey 2001] é, na verdade, uma implementação de um *framework* conceitual ou, como os próprios autores dizem, um *kit de ferramentas* para dar suporte ao desenvolvimento rápido de aplicações sensíveis ao contexto.

Este *framework* conceitual é composto de elementos que atuam desde a aquisição até o processamento e distribuição de informações de contexto às aplicações. Seus elementos são: *context widgets*, interpretadores (*interpreters*), agregadores (*aggregators*),

serviços (*services*) e descobridores (*discoverers*). A seguir, cada elemento será examinado em maiores detalhes.

Context widgets: são componentes de software empregados para fornecerem às aplicações acesso à informação de contexto, ou seja, são mediadores localizados entre a aplicação e seu ambiente operacional. Suas principais funções são:

8. *Fornecer uma separação de interesses*, escondendo da aplicação a complexidade dos sensores reais utilizados;
9. *Fornecer informações abstratas* de forma a alimentar a aplicação com apenas informações relevantes;
10. *Fornecer blocos de aquisição* de contexto reusáveis e personalizáveis que possam ser utilizados por uma variedade de aplicações sem a necessidade de serem alterados.

Interpretadores (*Interpreters*): são os responsáveis por elevar o nível de abstração de um certo "bloco" de contexto. Ou seja, um interpretador de contexto recolhe informações de uma ou mais fontes de contexto e produz uma nova unidade de informação contextual. Todos os interpretadores têm uma interface comum facilitando, assim, a identificação das capacidades que um determinado interpretador fornece, além de permitir a comunicação com o ambiente externo.

Agregadores (*Aggregators*): têm a função de reunir múltiplos pedaços de informação de contexto que estão logicamente relacionados e armazená-los em um repositório comum. De acordo com os autores, a necessidade de agregação surge devido principalmente à natureza distribuída da informação de contexto.

Serviços (*Services*): Nota-se que os três primeiros componentes citados (widgets, interpretadores e agregadores) são responsáveis pela aquisição de contexto e sua respectiva entrega às aplicações interessadas nesses dados. Porém, os serviços "são os componentes no framework que executam as ações em favor das aplicações" [Dey 2001]. Para os autores, estes serviços podem ser síncronos ou assíncronos, dependendo ou não da necessidade de se esperar uma resposta para que outras ações sejam executadas subsequentemente.

Descobridores (*Discoverers*): destinam-se a manter um registro de quais capacidades existem, em cada momento, no framework. Por esta razão, são considerados os componentes finais no framework conceitual proposto. Portanto, os descobridores devem conhecer quais widgets, interpretadores, agregadores e serviços estão atualmente disponíveis para uso pelas aplicações. A Figura 13.8 exibe o diagrama de objetos para as abstrações descritas anteriormente para o Context Toolkit.

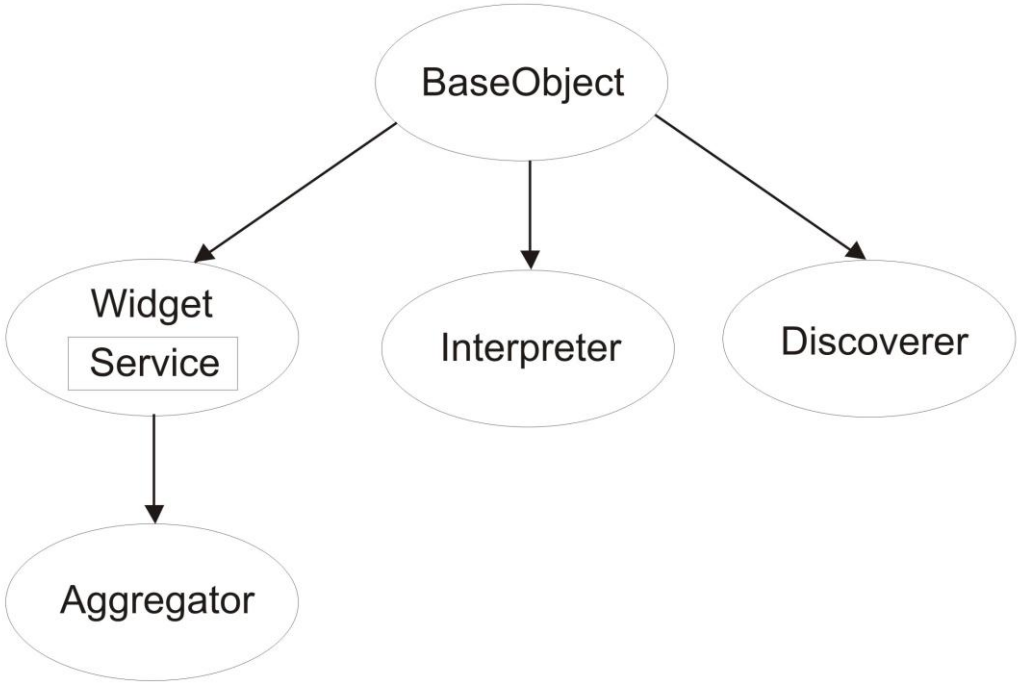


Figura 13.8. Diagrama de objetos para as abstrações do Context Toolkit.

Segundo este modelo proposto, "um número limitado de relacionamentos podem ser estabelecidos entre esses componentes. Widgets podem ser consultados ou usarem notificações para informar a seus clientes sobre mudanças ocorridas. Seus clientes podem ser aplicações, agregadores ou outros widgets. Agregadores, normalmente, atuam como pontes entre widgets e as aplicações. Interpretadores podem ser solicitados em qualquer estágio e por qualquer widget, agregador ou aplicação. Serviços são acionados primeiramente pelas aplicações (embora outros componentes possam também usá-los). E, finalmente, descobridores se comunicam com todos os componentes, adquirindo informação de widgets, interpretadores e agregadores e fornecendo estas informações às aplicações via notificações ou consultas." [Dey 2001].

Como já foi citado anteriormente, o Context Toolkit é uma implementação deste framework conceitual discutido nos parágrafos anteriores. Esta implementação foi desenvolvida em Java, embora, de acordo com seus autores, tenham sido usados mecanismos independentes de linguagem de programação.

É importante notar que, nessa implementação, cada componente (widget, interpretadores, agregadores e descobridores) é implementado com um único processo. No Context Toolkit, componentes diferentes podem estar distribuídos em processadores diferentes. Por esta razão, este framework é construído no topo de uma simples infraestrutura distribuída utilizando comunicações ponto-a-ponto.

Comunicação Distribuída

Para a comunicação distribuída, foi empregado um mecanismo de comunicação de objetos simples baseada em HTTP (*HyperText Transfer Protocol*) e XML (*eXtensible Markup Language*) para codificar as mensagens. A habilidade da comunicação distribuída, nesse caso, é encapsulada em um componente denominado BaseObject. Por isso, tanto os

widgets quanto os interpretadores são implementados como subclasses do BaseObject; porém, os agregadores são subclasses dos widgets. Esta hierarquia permite que todos esses componentes possuam habilidades de comunicação embutidas.

Processo de Inscrição para Recebimento de Contexto

Para que uma aplicação possa se inscrever em um *context widget*, ela deverá primeiramente adquirir um manipulador (*handle*) para o widget correspondente. Isso é feito contactando o descobridor e especificando qual tipo de informação de contexto é necessária. De posse do manipulador do widget desejado, pode-se usar sua instância do BaseObject para contactar o widget. Ao receber a comunicação, o BaseObject do widget decodifica a mensagem e a entrega ao widget propriamente dito. Este adiciona a aplicação em sua lista de aplicações interessadas em seu contexto juntamente com todos os dados para uma posterior comunicação (*handle* da aplicação, por exemplo). Desse modo, quando um widget recebe nova informação de contexto de seu sensor, ele recorre à sua lista de aplicações cadastradas para enviar o contexto apropriado a cada uma delas.

O contexto é enviado pelo widget usando sua instância do BaseObject que se omunicará diretamente com o BaseObject da aplicação. Então, o BaseObject da aplicação decodificará a mensagem enviada e chamará, em seguida, o método apropriado da aplicação.

Manipulando Eventos

Para se entender como os eventos são manipulados no Context Toolkit, deve-se observar que quando alguma informação de contexto é modificada, esta modificação é detectada pelo context widget responsável pela gerência deste pedaço de informação. Em seguida, o widget responsável gera uma marca de tempo, indicando o momento da alteração e, então, notifica todos os seus clientes inscritos em sua lista de interessados naquela informação de contexto (widgets, agregadores e aplicações).

Frequentemente, a nova informação é modificada pelos interpretadores encontrados pelo caminho entre o widget e o cliente. Pelo fato de o widget e seus clientes não necessariamente estarem executando na mesma máquina, os eventos são enviados através da infraestrutura distribuída, como mensagens XML sobre o protocolo HTTP.

Mecanismo de Descoberta

É importante observar que o mecanismo de descoberta usado no Context Toolkit é centralizado. Ou seja, é empregado apenas um único descobridor e não uma federação de descobridores, como discutida no *framework* conceitual. O processo todo funciona da seguinte maneira.

Quando iniciados, todos os widgets, agregadores e interpretadores se registram com o descobridor em um endereço de rede e uma porta conhecidos por todos os componentes. Logo após, o descobridor verifica se cada componente registrado está funcionando adequadamente. Caso durante este teste algum componente falhe, o componente é removido da lista de registros e outros componentes que registraram interesse neste componente específico são notificados. Além disso, sempre que algum componente é desativado manualmente, eles próprios notificam o descobridor indicando que já não podem mais serem utilizados.

Serviços de Contexto

No Context Toolkit, os serviços são incorporados nos widgets. Portanto, além de um widget ser responsável pela coleta de dados de um sensor, ele também tem a habilidade de executar ações no ambiente em que está inserido. Por exemplo, um Widget Temperatura pode não apenas indicar a temperatura corrente emitida por um sensor sob sua responsabilidade, como também alterar a quantidade de vapor da caldeira em que está inserido seu sensor.

Uma Aplicação-Exemplo

A Figura 13.9 mostra o diagrama arquitetural para a aplicação Guia Turístico Móvel (*Mobile Tour Guide*). Esta é uma das aplicações sensíveis ao contexto mais comuns e permite a um visitante percorrer ambientes internos e externos desconhecidos. À medida em que o visitante se move, a aplicação exibe na tela do dispositivo móvel a sua localização corrente em um mapa e exibe ainda informações sobre os locais interessantes que estão próximos a ele.

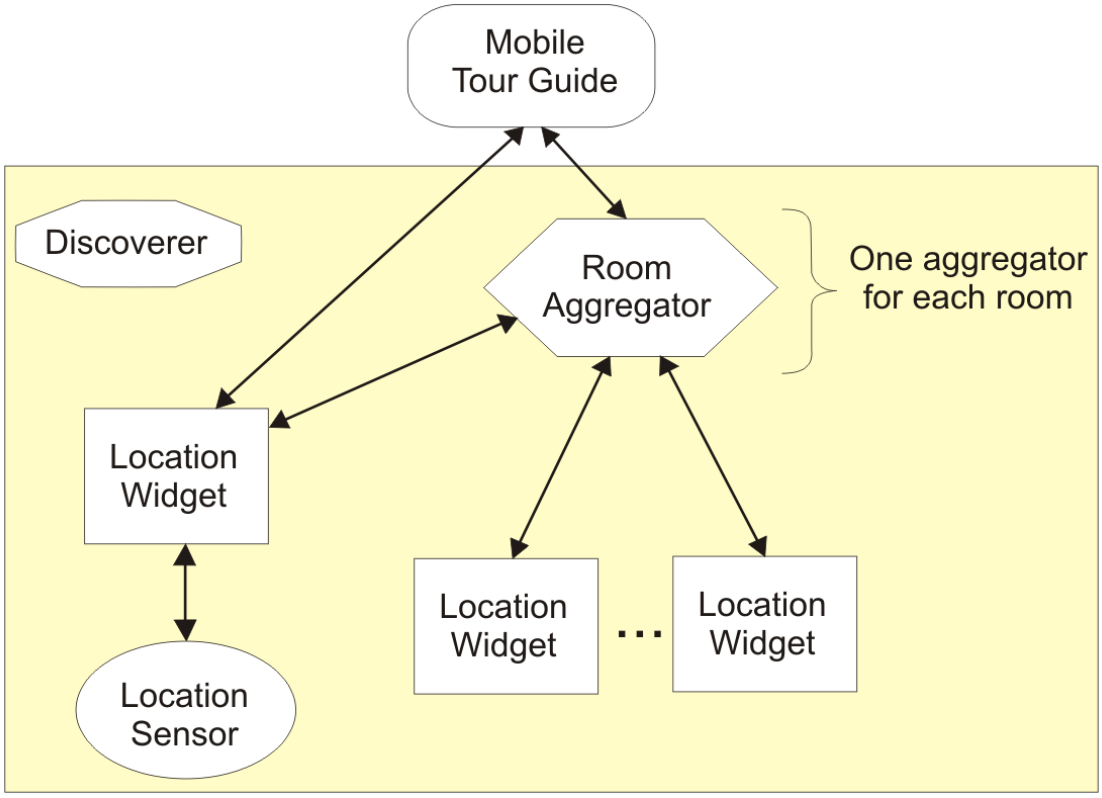


Figura 13.9. Diagrama arquitetural para a aplicação Guia Turístico Móvel.

Segundo os autores [Dey 2001], para esta aplicação, deve-se assumir que ela deverá ser executada em um ambiente interno (*indoor space*) e que este ambiente possui um sistema de localização que relata atualizações de localização para um servidor centralizado. O Location Widget exibido no diagrama arquitetural é construído sobre um servidor de localização centralizado. Cada *Agregador de Sala* ou *Room Aggregator* (um para cada sala) se registra no Location Widget para ser notificado quando um usuário entrar ou sair de uma sala correspondente. Os agregadores de sala também se registram nos Exhibit Widgets (Widgets de Exibição de Imagens dos ambientes) em suas respectivas salas, assim ficam sabendo quais imagens a serem exibidas estão disponíveis em um dado instante. Cada Exhibit Widget contém informações sobre um ícone (representação de um objeto

real) na rota. A aplicação se registra no Location Widget para atualizações de localização para seus usuários. Quando recebe uma atualização, ela atualiza seu mapa e consulta o Agregador de Sala correspondente à localização corrente do usuário para verificar a lista de ícones de exibição disponíveis, se existirem. Em seguida, exibe o ícone correspondente ao usuário e também se registra junto ao Agregador de Sala para ser notificado sobre quaisquer mudanças na lista de ícones de exibição.

13.4.2. ConBus

O ConBus [Sá 2010] é uma arquitetura para provisão de contexto em dispositivos móveis. Conforme ilustrado na Figura 13.10, ela é constituída por três camadas: Camada de Sensores, *Middleware* ConBus e Camada de Aplicação.

O *middleware* ConBus, que implementa a parte intermediária da arquitetura, executa, de forma co-localizada com as aplicações, no dispositivo móvel. Dessa forma, mais de uma aplicação, executando no mesmo dispositivo móvel, poderá utilizar a mesma instância do *middleware* ConBus para obter dados de contexto desejados, desde que sejam instalados os módulos de sensoriamento adequados a cada uma delas. Este *middleware* oferece às aplicações sensíveis ao contexto, através de uma interface de acesso padronizada, acesso a todas as informações contextuais providas pelos sensores acoplados ao *middleware*. Deve-se observar que, visando minimizar o tempo e o custo de se desenvolver novas aplicações sensíveis ao contexto, um desenvolvedor de uma aplicação móvel poderá, ainda, acoplar ao ConBus outros módulos de sensoriamento existentes sempre que houver necessidade, desde que esses módulos de sensoriamento tenham sido desenvolvidos de acordo com esta arquitetura.

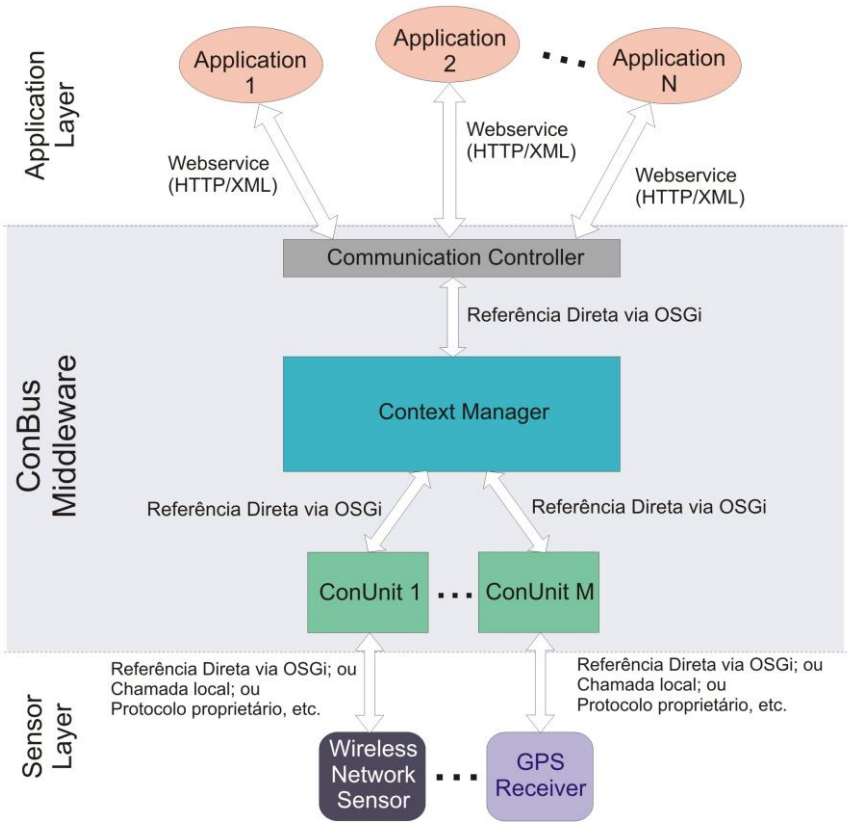


Figura 13.10. A arquitetura ConBus – Modelo Geral.

Camada de Sensores

A Camada de Sensores (*Sensor Layer*), ilustrada na Figura 4.8, é constituída por todas as fontes de contexto (sensores) que fornecem informações ao *middleware* ConBus.

Em princípio, além dos *sensores físicos*, como aqueles que coletam dados sobre temperatura, localização, luminosidade, nível de energia, etc; outros tipos de sensores, denominados *sensores lógicos*, podem ser incorporados à arquitetura ConBus. Estes sensores podem coletar dados de contexto, como os compromissos armazenados em agendas eletrônicas, preferências de usuários, páginas Web visitadas, dentre outros.

Middleware ConBus

O *middleware* ConBus é a camada intermediária que gerencia todas as informações contextuais coletadas pelas fontes de contexto da Camada de Sensores e que, posteriormente, são fornecidas a todas as aplicações sensíveis ao contexto que fazem uso das mesmas. Esta camada é composta pelos seguintes componentes: o *Gerenciador de Contexto* (*Context Manager*), responsável pelo controle de todos os recursos oferecidos pelo *middleware*, ou seja, o controle de todas as informações coletadas pelos sensores acoplados ao ConBus, bem como a responsabilidade de disponibilizá-las adequadamente às aplicações sensíveis ao contexto. Além do Gerenciador de Contexto, há também os ConUnits que interligam, de forma padronizada, as fontes de contexto ao ConBus. há também o Controlador de Comunicações (*Communication Controller*) cuja função principal é permitir a comunicação entre o *middleware* ConBus e as aplicações sensíveis ao contexto.

Outros componentes importantes são os Interpretadores de Contexto (*Context Interpreters*), que realizam tarefas importantes para a arquitetura, inferindo novas informações de contexto a partir de dados de contexto de mais baixo nível, como, por exemplo, a indicação de que um usuário se encontra ou não em uma reunião, inferida a partir de sua localização, data e hora correntes e as informações de sua agenda de compromissos.

Além dos componentes citados anteriormente, há ainda o framework OSGi [Alliance 2009] cuja função é permitir o acoplamento de novos ConUnits e Interpretadores de Contexto ao Context Manager. O OSGi permite ainda que ConUnits ou Interpretadores de Contexto sejam instalados, desistalados, iniciados, desligados, e até, atualizados dinamicamente sem a necessidade de reiniciar a execucão de todo o *middleware* ConBus. Isto permite que o próprio ConBus seja sensível ao contexto e, sempre que necessário, possa instalar ou desistalar, iniciar ou desligar ConUnits ou Interpretadores de Contexto, visando poupar recursos computacionais, como, energia, uso de CPU e memória, etc. Além disso, pode-se realizar, remotamente e de forma automatizada, atualizações em ConUnits e/ou Interpretadores de Contexto.

Camada de Aplicação

A Camada de Aplicação (*Application Layer*) é constituída por todas as aplicações sensíveis ao contexto que utilizam as informações contextuais fornecidas pelo *middleware* ConBus. Para usufruir das informações de contexto providas pelo ConBus, uma aplicação necessita apenas conhecer a URL (neste caso, http://localhost:8585) da instância do *middleware* ConBus que está sendo executada no dispositivo móvel correspondente e, em seguida, invocar os métodos desejados com os devidos parâmetros oferecidos pela interface de acesso ao contexto, usando, para isso, interfaces síncronas e assíncronas providas pela API

das Aplicações (*Application API*). Esta API define um *Web Service* local baseado em REST [O'Reilly 2009], através do qual as aplicações podem enviar requisições para a instância do ConBus que se encontra em execução.

A comunicação síncrona ocorre da seguinte forma: como dito anteriormente, o ConBus exporta um *Web Service* local (i.e., associado ao endereço *localhost*) em uma porta pré-definida (a padrão é 8585) para o qual as aplicações podem enviar requisições utilizando o comando GET e informar, como parâmetros, o nome da variável de contexto na qual estão interessadas (LOCATION, TEMPERATURE, ENERGYLEVEL, etc.) e a entidade à qual essa variável está relacionada (ex. CARLOS14352, SALA113, NokiaE51-1213, etc.). Como resposta, recebem a informação de contexto correspondente, representada em um formato XML predefinido.

A comunicação assíncrona acontece de forma semelhante, porém, deve-se enviar requisições utilizando-se o comando POST, seguido pela palavra *Assinc* e informando, em seguida, o nome da variável de contexto desejada juntamente com uma *expressão* que contenha uma condição ou restrição e, ainda, o nome da entidade à qual essa variável está relacionada (por exemplo, ENERGYLEVEL <= 25%, iPhone2132) para que a instância do ConBus em execução envie as informações contextuais das quais a aplicação está interessada.

Nota-se que, à semelhança com o Context Toolkit, apresentado na Subseção anterior, apesar da instância do ConBus e as aplicações estarem executando localmente no mesmo dispositivo móvel, a comunicação entre eles ocorrerá via interface de rede, neste caso, um *Web Service* local; segundo os autores, isto foi proposto para tornar o uso do ConBus independente da linguagem na qual a aplicação foi desenvolvida.

13.4.3 MoCA - Mobile Collaboration Architecture

A arquitetura MoCA [Sacramento 2004] [MoCATeam 2005] auxilia o desenvolvimento de aplicações colaborativas móveis através de serviços de coleta, agregação e difusão de informações de contexto computacional e de localização. A MoCA consiste de serviços de provisão contexto que disponibilizam para as aplicações o contexto da rede e do dispositivo, de APIs de comunicação e um framework (PrFw) para auxiliar a integração das aplicações com tais serviços através de uma comunicação síncrona ou assíncrona (baseada em eventos). Uma visão geral dos serviços da MOCA é ilustrada na Figura 13.11.

No projeto da MoCA, é definida uma separação de tarefas em serviços distintos, onde cada serviço implementa uma funcionalidade específica para a coleta ou divulgação do contexto. Por um lado, essa decisão de projeto favoreceu a flexibilidade no uso dos serviços, pois o desenvolvedor pode utilizar somente aqueles de que realmente necessita. Além disso, essa abordagem permite que novos serviços possam ser incorporados à arquitetura de forma modular, tornando os serviços do núcleo da MoCA mais escaláveis por não sobrecarregá-los com a implementação das funcionalidades envolvidas na provisão dos mais diversos tipos de informação de contexto (e.g, contexto computacional, contexto pessoal).

Por outro lado, essa decisão acarreta um forte acoplamento entre os serviços de provisão de contexto que precisam interagir uns com os outros para desempenhar suas funções. Por exemplo, o LIS usa no cálculo da inferência da localização as informações de RSSI providas pelo CIS, que, por sua vez, foram coletadas pelos Monitores executando nos

dispositivos móveis dos usuários. Esse acoplamento configura uma dependência de execução entre os serviços. Ou seja, para utilizar o LIS, o CIS deve estar executando. Para auxiliar a implantação e uso desses serviços, é descrito em [MoCATeam 2005] um documento (MoCA Overview) que discute as questões de dependências de execução e implementação entre as APIs e serviços, e os passos necessários para desenvolver uma aplicação baseada na MoCA.

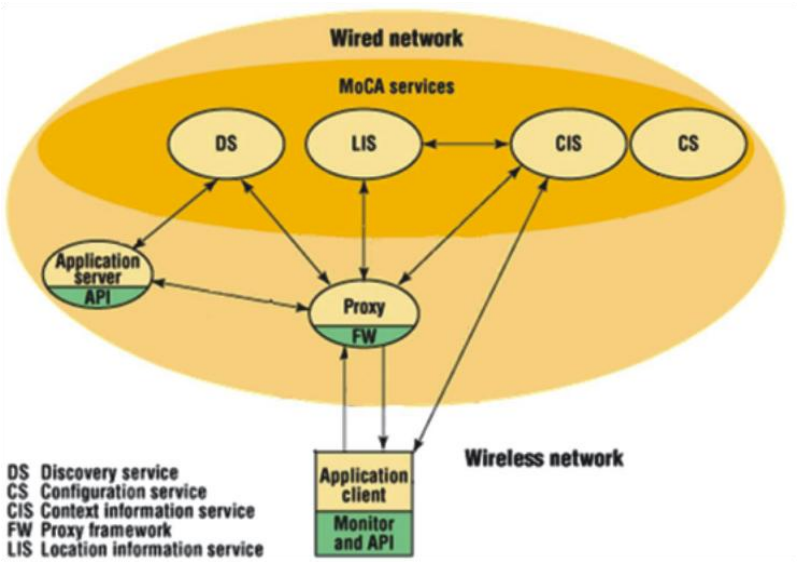


Figura 13.11. Arquitetura MoCA

Os serviços de provisão de contexto da MoCA são brevemente descritos nas subseções seguintes. Maiores detalhes sobre como esses interagem com os demais serviços da arquitetura podem ser encontrados em [Sacramento 2004] [MoCATeam 2005]. O CIS, o MONITOR e as APIs de comunicação síncrona e assíncrona (baseadas em eventos) constituem parte da contribuição deste trabalho. O serviço de localização (LIS) que foi o principal elemento usado no estudo de caso para o controle de privacidade, no entanto, foi resultado de uma dissertação de mestrado [Nascimento 2005].

13.4.3.1. Monitor

O Monitor coleta informações do dispositivo móvel e da rede sem fio e as envia periodicamente para o CIS na rede fixa. Tais informações (na forma de uma lista de variáveis) podem ser classificadas em duas categorias, como a seguir.

Informações coletadas da rede sem fio:

- intensidade do sinal e endereço MAC de todos os Access Points (APs) IEEE 802.11 que estão no raio de cobertura do dispositivo;
- endereço MAC e intensidade do sinal do AP corrente.

Informações coletadas sobre a configuração e os recursos disponíveis do dispositivo móvel:

- taxa de uso da CPU (em %);

- memória disponível (em Kbytes);
- endereço MAC e IP/Máscara;
- nível de energia disponível (em %).

Além dos envios periódicos das informações supracitadas, o MONITOR também as enviará quando for detectada uma mudança no endereço IP ou AP corrente, já que isso caracteriza um *roaming* (*handover*) e, portanto, representa uma mudança do estado/contexto corrente do dispositivo que possivelmente é de interesse para as aplicações.

As informações coletadas pelo MONITOR podem ser utilizadas para diferentes propósitos, como, por exemplo, implementar uma adaptação dinâmica das aplicações em função do nível de energia ou da memória disponível.

De uma forma geral, o serviço de sensoriamento é um componente chave para o desenvolvimento de qualquer arquitetura de provisão de contexto. No entanto, a implementação deste serviço geralmente demanda um grande esforço de programação por causa da complexidade em obter as informações acerca da utilização dos recursos de cada plataforma de hardware (e.g., iPAQs, Palms, Laptops), i.e., como obter a taxa de uso da CPU, bateria, ou, obter informações sobre cada possível interface de rede disponível.

Como exemplo da complexidade envolvida no desenvolvimento deste serviço, vale destacar que, às vezes, a implementação da coleta de um contexto específico (e.g., uso da CPU ou intensidade de sinal dos APs 802.11) para dispositivos de uma mesma plataforma e de um mesmo fabricante, como, por exemplo, iPAQs da HP, pode variar de acordo com a série do produto ou versão do sistema operacional.

13.4.3.2. Serviço de informação de contexto

O *Context Information Service* (CIS) é um serviço que recebe, processa e divulga para as aplicações interessadas os dados de contexto recebidos de diversos monitores. Através das APIs de comunicação, Communication Protocol e Event-based Communication Interface (ECI), a aplicação pode interagir com o CIS de duas formas: por meio de uma comunicação síncrona (requisição/resposta) para obter o valor de uma dada variável de contexto ou por meio de uma comunicação assíncrona, baseada em eventos, para ser notificada sobre uma eventual mudança de valor de alguma variável de contexto. Através das notificações, as aplicações podem manter-se a par das mudanças do estado corrente do dispositivo móvel (e.g., memória livre, nível de energia) ou da rede sem fio (e.g., variação do RSSI, AP corrente).

Para ser notificada, a aplicação deve se registrar no CIS (através de uma *subscription*) passando como parâmetro um tópico ou assunto (*Subject*) e, opcionalmente, uma expressão (baseada no padrão SQL92) para filtrar os eventos de seu interesse. Através da API ECI, o CIS recebe e registra a subscription e, eventualmente, publica um evento que satisfaz as propriedades da assinatura de interesse de uma aplicação. Para o CIS, o tópico (i.e., o Subject) é o endereço MAC do dispositivo e as propriedades especificam

uma expressão sobre o estado das variáveis de contexto deste dispositivo (e.g., Roaming, FreeMemory) nas quais a aplicação está interessada.

Uma subscription, por exemplo, poderia ser Subject = {"02:DA:20:3D:A1:2B"}, Properties = {"roaming = True" OR "FreeMem < 15%" OR "CPU > 90%" OR (("OnLine = False") AND ("DeltaT > 15s"))}. A aplicação recebe então uma notificação sempre que a condição informada na expressão for satisfeita. Essa notificação descreve o valor corrente das tags/variáveis de contexto do dispositivo.

A aplicação também pode ser notificada quando a condição especificada através da expressão voltar a ser invalidada, i.e., a ter o valor "falso" (evento conhecido também como negação da expressão). Essa funcionalidade é essencial para as aplicações que implementam algum tipo de adaptação baseada em contexto. Por exemplo, após ser notificada que um determinado estado de alguma tag de contexto do dispositivo está abaixo do limiar aceitável (e.g., nível de energia abaixo de 10%), a aplicação pode desempenhar alguma função, que possivelmente tem alto custo computacional, para lidar com a situação corrente, como, por exemplo, ativar uma política de cache, compactar ou remover parte dos dados enviados para o cliente executando no dispositivo em questão. Por isto, a aplicação também deve ser notificada quando o referido estado do nível de energia voltar a ficar acima do limiar aceitável. Essa funcionalidade permite que a aplicação pare de executar a adaptação.

Em linhas gerais, o CIS recebe as informações enviadas pelos Monitores, faz uma análise sintática no conteúdo da mensagem e extrai os dados que descrevem o contexto do dispositivo e da rede. Em seguida, ele publica tais dados, usando a API do serviço de eventos, para as aplicações que tenham se registrado como interessadas em alguma mudança de estado do dispositivo ou da rede. A implementação do CIS, da API ECI e Communication Protocol estão disponíveis para download em [MoCATeam 2005a] [MoCATeam 2005b] [MoCATeam 2005c].

13.4.3.3. Serviço de inferência de localização

O *Location Inference Service* (LIS) infere e disponibiliza a localização simbólica dos dispositivos móveis em áreas cobertas por Access Points (APs) de redes IEEE 802.11. Para isso, ele utiliza a intensidade de sinais RSSI coletados e divulgados pelos monitores e CIS, respectivamente.

Conforme descrito em [Nascimento 2005], a inferência é feita a partir da comparação da similaridade entre o padrão de sinal de RF atual (representado por um vetor, onde cada elemento deste vetor corresponde ao sinal de um AP "audível" ao dispositivo) e o padrão (vetor RSSI) medido anteriormente em pontos pré-definidos em uma região com cobertura de rede IEEE 802.11. Portanto, em uma primeira etapa (fase de calibração), as amostras de vetores RSSI são coletadas em pontos de referência bem definidos na(s) área(s) de interesse, e armazenadas em um banco de dados do LIS. Considerando que os sinais de rádio são suscetíveis à intensa variação e interferência, a precisão da inferência da

localização depende fortemente do número de APs, do número de pontos de referência escolhidos e do número de amostras de vetores RSSI coletados.

O LIS atende, principalmente, aplicações que necessitam conhecer a posição de um dispositivo em termos de regiões simbólicas (ao invés de coordenadas), onde tais regiões são áreas geográficas não menores do que 1 a 4m2. Devido à flutuação intrínseca no sinal de rádio, a inferência da localização baseada em RSSI não é capaz de oferecer resultados com maior precisão. Entretanto, acredita-se que a precisão da localização obtida pelo LIS é suficiente para uma grande classe de aplicações sensíveis à localização. Ao invés de coordenadas físicas (e.g., latitude/longitude), várias aplicações sensíveis à localização necessitam apenas saber a localização simbólica de usuários e dispositivos.

No LIS, regiões simbólicas podem ser estruturadas hierarquicamente e são representadas como qualquer área geográfica com uma semântica (ou identificação) bem definida, tal como uma sala específica, corredor, andar de um prédio, prédio, etc. Essa funcionalidade permite que as aplicações LBS obtenham a informação de localização em diferentes granularidades. Para tanto, o LIS implementa um modelo hierárquico de localizações, como aqueles discutidos nos trabalhos [Ranganathan 2004] [Roth 2003].

Neste modelo (naturalmente representado como uma árvore), regiões simbólicas podem ser compostas por outras regiões simbólicas menores (aninhadas). Nesta hierarquia, os nós folhas são definidos como regiões simbólicas atômicas, as menores unidades de localização que podem ser reconhecidas pelo LIS. Os ancestrais de um nó folha da hierarquia de regiões representam as demais regiões simbólicas as quais um dado dispositivo pode estar implicitamente associado.

Por exemplo, dada a hierarquia “PUC-Rio/Prédio RDC/5o Andar/Sala 501”, o LIS pode inferir que um dado dispositivo se encontra no “Prédio RDC” caso a localização corrente desse dispositivo seja “Sala 501”. O LIS oferece uma interface através da qual cada aplicação pode criar e gerenciar sua própria topologia de regiões simbólicas baseada nas regiões atômicas definidas pelo administrador do serviço.

13.4.4 Hydrogen

O projeto Hydrogen [Hofer 2003] propõe uma arquitetura e uma implementação de um *framework* para facilitar o desenvolvimento de aplicações móveis sensíveis ao contexto. Ao contrário de algumas outras infraestruturas com a mesma finalidade, esta abordagem é totalmente executada no próprio dispositivo móvel.

O *framework* é definido em uma arquitetura composta por três camadas: camada de adaptadores, camada de gerência e camada de aplicação. A Figura 13.12 exibe o diagrama arquitetural do Hydrogen.

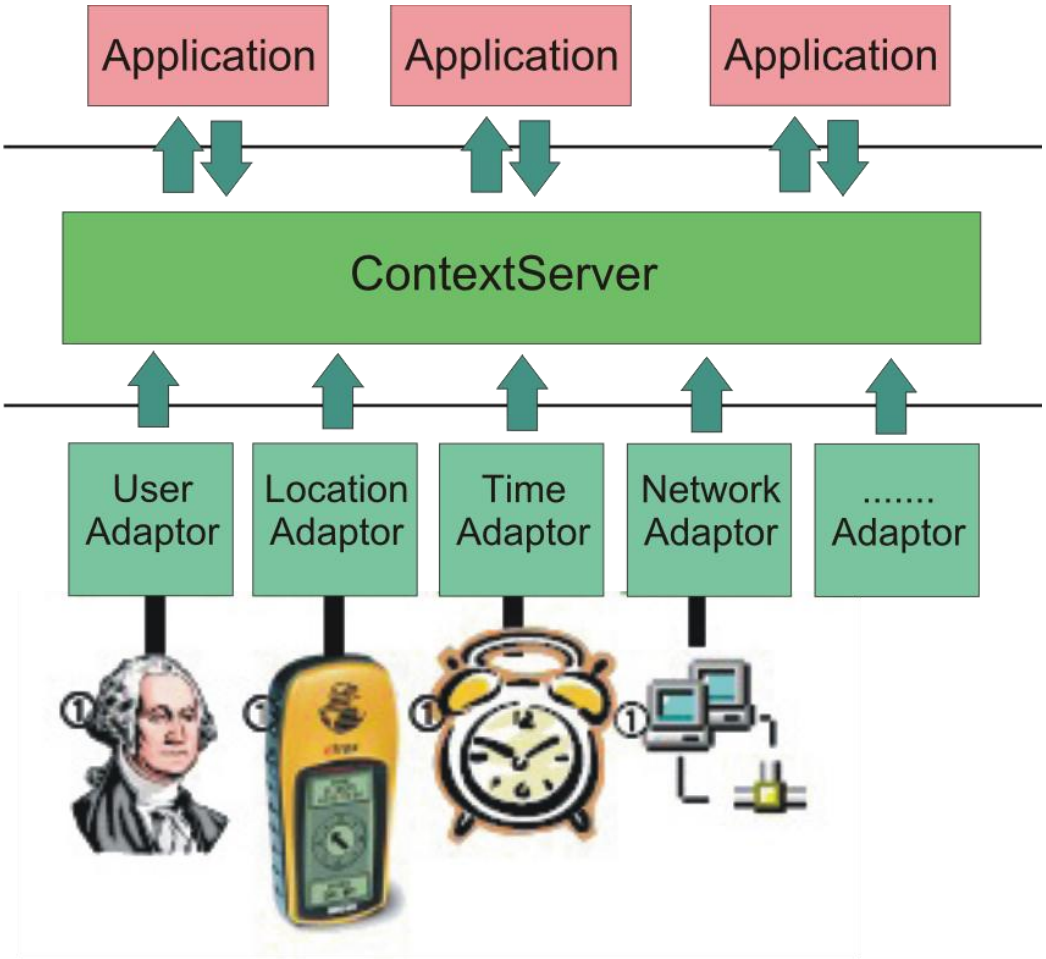


Figura 13.12. Diagrama arquitetural do Hydrogen.

A Camada de Adaptadores (*Adaptor Layer*) é a responsável pela obtenção de informações de contexto dos sensores. Nesta camada é possível ainda enriquecer os dados obtidos dos sensores através de algum tipo de processamento. Por exemplo, a partir de coordenadas GPS (latitude, longitude e altitude), o adaptador responsável por obter estes dados poderia ainda inferir um endereço postal, como o nome da rua, bairro e número da localidade indicada pelas coordenadas GPS.

A Camada de Gerenciamento (*Management Layer*) tem a função de armazenar toda a informação de contexto obtida pelos sensores (mediante o intermédio dos adaptadores) sobre o ambiente atual do dispositivo móvel e disponibilizá-los às aplicações interessadas. Seu principal componente é o Servidor de Contexto (*ContextServer*). O Servidor de Contexto é um objeto executável Java. É acessível através de uma porta de comunicação definida pelo usuário, onde, após ser iniciado, ficará aguardando comandos enviados pelos adaptadores e aplicações. Como foi visto anteriormente, nessa arquitetura, os adaptadores fornecem informações de contexto ao Servidor de Contexto que podem ser consultadas, de forma síncrona ou assíncrona, pelas aplicações.

Uma carência da própria arquitetura é a inexistência de módulos ou componentes na camada de gerência para realizar processamento de alto nível sobre as informações de contexto fornecidas pelos adaptadores. Ou seja, esta camada não oferece às aplicações serviços como a interpretação, agregação e classificação de contexto. De modo que praticamente o único serviço oferecido às aplicações é o armazenamento e disponibilização de contexto provenientes dos adaptadores.

Finalmente, a Camada de Aplicações (*Application Layer*) engloba, obviamente todas as aplicações sensíveis ao contexto que utilizam as informações oriundas das camadas anteriores.

Além da arquitetura de um *framework* para facilitar o desenvolvimento de aplicações móveis sensíveis ao contexto, os autores também propõem uma arquitetura para o contexto, isto é, para representar o contexto a ser manipulado por este *framework*. Há, basicamente, dois tipos de contexto nesta arquitetura: *contexto local* e *contexto remoto*. O contexto local é, naturalmente, toda informação de contexto obtida no próprio dispositivo, enquanto o contexto remoto significa qualquer informação de contexto fornecida por outros dispositivos logicamente relacionados e que podem se comunicar com o dispositivo em questão.

Toda informação de contexto, seja local ou remota, é, na verdade, uma coleção de objetos de contexto (*ContextObject*). Tais objetos podem ser: contexto temporal (*TimeContext*), contexto do dispositivo (*DeviceContext*), contexto de rede (*NetworkContext*), contexto de localização (*LocationContext*), contexto de usuário (*UserContext*), dentre outros.

Segundo os autores, o principal benefício desta arquitetura é o fato de as três camadas do *framework* estarem localizadas no dispositivo móvel. Esse fato, de acordo com os autores, torna a abordagem robusta contra possíveis e frequentes desconexões de rede. Porém, isto pode ser apenas parcialmente verdadeiro, visto que se uma aplicação faz uso de contexto remoto, a robustez contra desconexões de rede poderá não acontecer se as informações remotas forem essenciais para o funcionamento adequado da aplicação em questão.

13.5. Considerações Finais

Tanto a Computação Móvel quanto a Computação Ubíqua/Pervasiva têm muito a ganhar com a evolução da Computação Sensível ao Contexto, visto que para se desenvolver aplicações móveis e pervasivas mais “inteligentes” e inovadoras é necessário obrigatoriamente empregar, de alguma forma, as técnicas e soluções propostas pela Computação Sensível ao Contexto.

Além disso, com a popularização de dispositivos móveis com cada vez mais sensores embutidos, como acelerômetros, receptores GPS, giroscópios e bússulas, os desenvolvedores de aplicações móveis e pervasivas podem facilmente lançar mão da sensibilidade ao contexto em seus projetos.

13.6. Referências

Mateus, G. R. and Loureiro, A. A. F. (1998) "Introdução à Computação Móvel", In: 11a Escola de Computação, COPPE/Sistemas, NCE/UFRJ., Brazil.

Weiser, M. (1991) "The computer for the twenty-first century", In: M., Scientific American, pp. 94–100, September.

Loke, S. (2006) "Context-Aware Pervasive Systems", In: Auerbach Publications, Boston, Ma, USA.

Ryan, J. and Pascoe, N. and Morse, D. (1997) "Enhanced reality fieldwork: the context-aware archaeological assistant," In: Gaffney, V., Leusen, M. v. and Exxon, S. (Eds.).

Abowd, G. D. and Dey, A. K. and Brown, P. J. and Davies, N. and M. Smith and P. Steggles (1999) "Towards a better understanding of context and context-awareness," In: HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing. Springer-Verlag, pp. 304–307, London, UK.

Schilit, B. N. and Theimer, M. M. (1994) "Disseminating active map information to mobile hosts," In: IEEE Network, 8(5), pages 22-32. [Online]. Available: http://schilit.googlepages.com/ams.pdf.

Want, R. and Hopper, A. and Falcão, V. and Gibbons, J. (1992) "The active badge location system," In: Olivetti Research Ltd. (ORL), 24a Trumpington Street, Cambridge CB2 1QA, Tech. Rep. 92.1. [Online]. Available: iteseer.nj.nec.com/want92active.html

Dictionary. (2011) "Free online dictionary of computing," In: [Online]. Available: http://dictionary.reference.com.

Gu, T. and Pung, H. K. and Zhang, D. Q. (2005) "A service-oriented middleware for building context-aware services," In: Journal of Network and Computer Applications, vol. 28, no. 1, pp. 1–18, January. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2004.06.002.

Bill N. A. and Schilit N. and W. R. (1994) "Context-aware computing applications," In: Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, 8(5), pages 85-90, IEEE Computer Society. [Online]. Available: http://schilit.googlepages.com/publications.

Chen, G. and Kotz, D. (2000) "A survey of context-aware mobile computing research," In: Technical Report TR2000-381 - Dartmouth College. [Online]. Available: http://www.cs.dartmouth.edu/reports/TR2000-381.pdf.

Salber, D. and Dey, A. K. and Abowd, G. D. (1999) "The context toolkit: aiding the development of context-enabled applications," In: CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems. New York, NY, USA: ACM, pp. 434–441.

Dey, A. and Salber, D. and Abowd, G. (2001) "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications". [Online]. Available: http://citeseer.ist.psu.edu/dey01conceptual.html

Sá, M. P. de (2010) "Conbus: Uma plataforma de middleware de integração de sensorespara o desenvolvimento de aplicações móveis sensíveis ao contexto," In: Dissertação de mestrado, Instituto de Informática – Universidade Federal de Goiás (INF/UFG), Brasil.

Alliance, O. (2009) “Osgi - the dynamic module system for java.” In. [Online].Available: http://www.osgi.org/Main/HomePage.

O’Reilly. (2009) “Implementing rest web services: Best practices and guidelines.” In: [Online]. Available: http://www.xfront.com/REST-Web-Services.html.

Baldauf, M. and Dustdar, S and Rosenberg, F. (2007) “A survey on context-aware systems.” In: Int. J. Ad Hoc Ubiquitous Comput., vol. 2, no. 4, pp. 263–277.

Sacramento, V. and Endler, M. and Rubinsztejn, H. K. and Lima, L. S. and Gonçalves, K. and Nascimento, F. N. and Bueno G. A. (2004) “Moca: A middleware for developing collaborative applications for mobile users” In: IEEE Distributed Systems Online, vol. 5, no. 10, p. 2,

MoCATeam (2005) “Moca home page,” In: http://www.lac.inf.puc-rio.br/moca (Last visited: April 2007).

Nascimento, F. N. da C. (2005) “Um serviço para inferência de localização de dispositivos móveis baseado em redes ieee 802.11,” In: Dissertação de mestrado, Pontifícia Universidade Católica do Rio de Janeiro.

MoCATeam (2005a) “Context information service (cis) home page,” In: Laboratory for Advanced Collaboration, PUC-Rio, http://www.lac.inf.puc-rio.br/moca/cis/ (Last visited: April 2007).

MoCATeam (2005b) “Event-based communication interface (eci) home page,” In: Laboratory for Advanced Collaboration, PUC-Rio, http://www.lac.inf.puc-rio.br/moca/eventservice/ (Last visited: April 2007).

MoCATeam (2005c) “Communication protocol (eci) home page,” Laboratory for Advanced Collaboration, PUC-Rio, http://www.lac.inf.puc-rio.br/moca/event-service/ (Last visited: April 2007).

Ranganathan, A. and Al-Muhtadi, J. and Chetan, S and Campbell, R. and Mikunas, D. (2004) “Middle-Where: A Middleware for Location Awareness,” in Proc. of the International Middleware Conference, Toronto, ser. LNCS, H. Jacobsen, Ed., vol. 3231, Oct. 2004, pp. 397–416.

Roth, J. (2003) “Flexible positioning for location-based services,” In: IADIS International Journal of WWW/Internet, vol. I, no. 2, pp. 18–32, 2003.

Hofer, T. and Schwinger, W. and Pichler, M. and Leonhartsberger, G. and Altmann, J. and Retschitzegger, W. (2003) “Context-awareness on mobile devices - the hydrogen approach,” In: System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on, 2003.

Capítulo 14

Introdução à Produção de Material Digital Acessível

Maludiane Nascimento[1], Társio Cavalcante[1] e Romero Freire[1]
Instituto Federal de Educação, Ciência e Tecnologia Baiano (IF Baiano)
, Rua Barão de Camaçari, 118, Barão de Camaçari – 48.110-000 – Catu – BA – Brasil
maludiane.nascimento, tarsio.cavalcante{@catu.ifbaiano.edu.br} e
romero.freire@ifbaiano.edu.br

Apresentador do Curso: Maludiane Nascimento

Resumo

A produção de material didático acessível é uma das maneiras de promover a inclusão do indivíduo ao meio virtual, por suprir a lacuna presente nos documentos digitais não acessíveis. A confecção de documentos digitais que atendem aos requisitos básicos dispostos na norma ANSI/NISO Z39.86-2005(DAISY3) para a criação de Livros Digitais Falados (do inglês, Digital Talking Book – DTB) será o tema principal desse minicurso. Um DTB refere-se a um documento digital que oferece comodidade principalmente aos leitores com deficiência visual ao manusear o conteúdo mais facilmente. As vantagens essenciais de um DTB é a navegação, conteúdo com correspondência em áudio e marcação de texto. As diretrizes contidas na norma DAISY3 defende a necessidade de fornecer qualquer tipo de conhecimento público com acessibilidade. Desta maneira, pode-se contribuir para a inclusão digital de pessoas com necessidades especiais. O objetivo do minicurso de Introdução à Produção de Material Digital Acessível tem como foco capacitar o participante a criar material digital em conformidade com os padrões de acessibilidade fixados na norma DAISY 3.

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

14.1 Dados gerais

14.1.2 Objetivos do curso

O curso de Introdução à Produção de Material Digital Acessível tem como objetivo geral capacitar o participante a criar materiais digitais acessíveis em conformidade com a norma ANSI/NISO Z39.86-2005(DAISY3).

14.1.3 Tipo do curso

O curso de Introdução à Produção de Material Digital Acessível terá uma abordagem teórica.

14.1.4 Tratamento dado ao tema

Inicialmente será realizada uma contextualização sobre a acessibilidade digital delimitando-a aos materiais digitais. Em seguida serão apresentados os resquisitos básicos para a criação de um livro digital falado estabelecidos pela DAISY 3 e os ambientes de desenvolvimento e execução dos DTBs.

14.1.5 Perfil desejado dos participantes

Profissionais e estudantes com interesse na área de acessibilidade digital, com conhecimento em informática básica.

14.1.6 Infra-estrutura física necessária para a apresentação

Equipamentos: microcomputador com sistema operacional *windows*; projetor multimídia; quadro-branco; pinceis; apagador; caixa de som. *Softwares*: Tocador MecDaisy, *Word* 7(com o *plugin Save – as Daisy)*, Editor Dirce, *DDR- Dorina Daisy Reader* e os pré-requisitos para instalação desses *softwares: Java 1.6 e* o ambiente .NET 3.0 instalado para os sistemas anteriores ao *Windows Vista.*

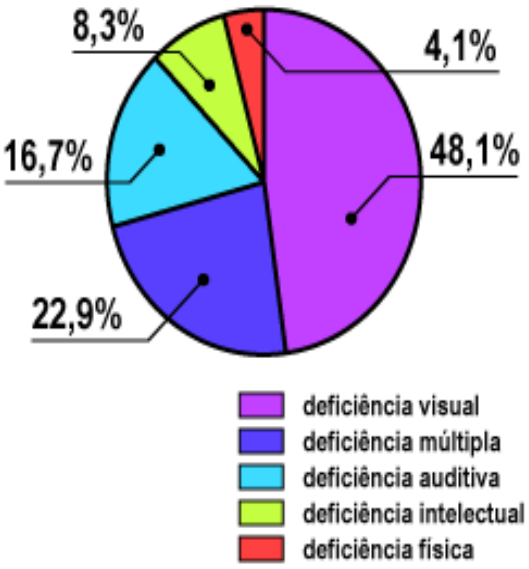
14.2 Introdução

O volume de dados produzido pelo homem tem aumentado substancialmente principalmente com o advento das Tecnologias da Comunicação e Informação (TIC). Desta maneira, o conhecimento extrapola os limites impostos pelo espaço físico da biblioteca, ou seja, este é transportado ao alcance de todos via a rede mundial de computadores. Uma forma de disseminação do conhecimento é através de conteúdos digitais multimídias.

O conteúdo digital de multimídia pode ser definido como um conjunto de informações textuais de imagens e de aúdio. Na área da educação, o conteúdo digital ganha maior relevância com a sua aplicabilidade como fonte de pesquisas. Os trabalhos acadêmicos digital, por exemplo, é um tipo de informação computadorizada que pode contribuir para democratização do saber. Além disso, o conteúdo digital pode ser utilizado apenas com o propósito informativo. Independentemente do objetivo em que o conteúdo digital seja utilizado, presume-se importante fornece-lo em um meio que agregue qualidade, tais como, a acessibilidade e usabilidade. Delimitando a acessibilidade ao aspecto digital, tem-se a necessidade de criar o conteúdo ou material digital de modo que respeite o direito equitativo de todos os humanos e promova o respeito pela dignidade

inerente das pessoas com deficiência. Em parte, o artigo 9 do Decreto Legislativo Nº 186, de 09 de Julho de 2008 declara o próposito da acessibilidade:

> *"Possibilitar às pessoas com deficiência viver de forma independente e participar plenamente de todos os aspectos da vida, os Estados Partes tomarão as medidas apropriadas para assegurar às pessoas com deficiência o acesso, em igualdade de oportunidades com as demais pessoas, ao meio físico, ao transporte, à informação e comunicação, inclusive aos sistemas e tecnologias da informação e comunicação, bem como a outros serviços e instalações abertos ao público ou de uso público, tanto na zona urbana como na rural. Essas medidas, que incluirão a identificação e a eliminação de obstáculos e barreiras à acessibilidade, serão aplicadas, entre outros, [...] a Informações, comunicações e outros serviços, inclusive serviços eletrônicos e serviços de emergência."*



Fonte: (IBGE, 2012)

Figura 14.1. População po tipo de deficiência no Brasil

Segundo o Instituto Brasileiro de Geografia e Estática (IBGE), o Brasil possuía em 2005 cerca de 14,5% de pessoas com deficiência (_IBGE, 2012). Nessa pesquisa constatou-se que aproximadamente 16,6 milhões de pessoas possuem uma deficiência visual e 5,7 milhões de pessoas tem uma deficiência auditiva. A Figura 14.1, exibe graficamente a distribuição do total de pessoas com deficiência no Brasil por tipo de deficiência. A deficiência visual é o tipo de deficiência mais comum com 48,1% , já a

deficiência física a menos comum com 4,1, a deficiência múltipla com 22,9%, a deficiência auditiva com 16,7% e a deficiência intelectual com 8,3%.

Cerca de 48,1 % dos brasileiros, incluindo crianças, adultos e idosos com deficiência visual, possuem o direito à educação e ao trabalho. As pessoas com deficiência visual total ou parcial normalmente utilizam leitores de tela ou ecrã, programas de computador que transformam o texto apresentado na tela em áudio. Alguns leitores de tela são gratuitos e agregam várias funcionalidades para aperfeiçoar o som transmitido e facilitar o manuseio por meio de atalhos de teclado. Para a execução desses programas com sucesso é preciso desenvolver conteúdos digitais acessíveis, isso envolve programas de computador e arquivo de texto. Outra parte da sociedade brasileira é composta por tetraplégicos, pessoas que tiveram uma lesão na medula e perderam todos os movimentos dos membros superiores e inferiores. Normalmente, os tetraplégicos necessitam de atalhos para, por exemplo, ir até um capítulo específico de um livro ou até mesmo utilizam emuladores de *mouse* (MULLER, 2001). Se essa pessoa estiver lendo um documento extenso que não ofereça um recurso para navegar entre os capítulos, ela terá dificuldades. E o que dizer das pessoas que não enxergam e nem ouvem? Como essas pessoas podem utilizar o livro digital sem desconforto?

Para contemplar a necessidade das pessoas com surdocegueira, o padrão DAISY, indica um modelo de Livro Digital Falado (do inglês, Digital Talking Book) que permite a conversão do texto em código para serem utilizados pela a linha braille. Uma linha braille é um dispositivo que pode reproduzir o texto presente no ecrã do computador (Figura 14.2).

Fonte: (GODINHO, 2012)

Figura 14.2. Linha *braille*

Um DTB ou LDF é um conjunto de arquivos eletrônicos preparados para apresentar a informação ao público-alvo por meio de métodos alternativos, isto é, voz humana ou sintetizada, terminal braile ou de fontes ampliadas. O DTB permite acessar a informação de maneira flexível e eficiente através da visão e audição. O diferencial do DTB, são os recursos de navegabilidade que permitem anotações e marcações de texto a partir de movimentos de teclas de atalhos ou do mouse, inclusive mudança de página .

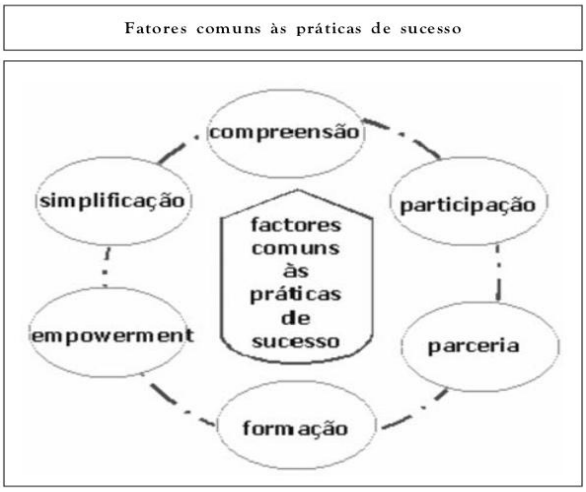
O consórcio DIASY iniciou suas atividades em 1996 e desde então tem expandido por todo o mundo com o objetivo de desenvolver tecnologia que promovam a igualdade de acesso à informação para as pessoas com deficiência visual e outras pessoas que tenham dificuldade no processo de leitura do material digital. Além desse público, o DAISY foi capaz de facilitar a leitura de algumas pessoas com deficiência física. Por não visar o lucro e sim a inclusão digital, a tecnologia DAISY é a mais apoiada em todo o mundo.

O consórcio criou um padrão para informação multimídia totalmente acessível, denominado DTB, para criação de livros ou qualquer material em formato digital. Através do livro digital falado, os usuários poderão pesquisar a informação como ela está disposta, por títulos ou subtítulos. Assim como um leitor pode ver a estrutura de um livro conforme a sequência da informação apresentada, uma pessoa com deficiência visual poderá ouvir em forma linear, palavra por palavra, na velocidade desejada, o conteúdo organizado de um livro. Portanto, o consórcio DAISY com o objetivo de tornar o acesso igual para todos, criou uma solução que pode ser desenvolvida a um custo relativamente baixo e que atinge o maior número de pessoas.

Outras iniciativas foram tomadas para conscientizar a população para o desenvolvimento acessível e transformar os portais do governo acessíveis, dentre essas a criação dos Núcleos de Acessibilidade Virtual (NAV). O NAV é fruto de um projeto apoiado pela RENAPI - Rede Nacional de Pesquisa e Inovação em Tecnologias Digitais (RENAPI). De acordo com testes e pesquisas realizadas, para a leitura de um site acessível é preciso utilizar corretamente as combinações de leitores de tela, navegadores e sistemas operacionais. Sendo este o objetivo do projeto Acessibilidade. O NAV é um dos núcleos que compõem o Grupo de Pesquisa Multidisciplinar em Informática Aplicada (MIA). O MIA agrega a expertise de projetos do Instituto Federal de Educação, Ciência e Tecnologia Baiano (IFBaiano), na perspectiva de formalizar as iniciativas de fomento a pesquisa e inovação. O objetivo do NAV é fomentar estudos e pesquisas sobre Acessibilidade Virtual, associado à Educação Inclusiva e Desenvolvimento de Software, na perspectiva de obter resultados que contribuam no processo de difusão do conhecimento, permitindo que um indivíduo, utilizando variados tipos de tecnologia de navegação (navegadores gráficos, textuais, especiais para cegos ou para sistemas de computação móvel), possa visitar qualquer site e compreender plenamente a informação disponibilizada. As pesquisas atuais do NAV concentram-se em Tecnologia Assistiva, Acessibilidade WEB e Inclusão e Necessidades Especiais. Outros iniciativas focalizaram o desenvolvimento de material digital acessível visto que a maioria são carentes de requisitos imprescidíveis a boa execução pelos leitores de tela. Sendo assim, para formalizar os resquisitos necessários para construir um documento acessível o padrão internacional *Digital Accessible Information System* (DAISY) fixou diretrizes para determinar o material digital acessível com qualidade. Os requisitos exigidos são navegabilidade, apresentação da informação em texto e em audio, estrururado, pesquisa anotações.

Portanto, a disponibilização do conteúdo digital acessível promove a inclusão digital e o direito a educação, ao acesso à informação e ao mercado de trabalho. Além desse atributo de qualidade de um material digital é a usabilidade. A usabilidade deve contemplar a inteligibilidade, apreensibilidade, operacionalidade e atratividade do produto (NBR 9126/1, 2003) e (NBR 9241/11, 2002). A usabilidade é uma medida para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso. A eficácia é a acurácia e completude com a qual o usuário, pessoa que interage com o software, alcança objetivos específicos. Já a eficiência é definida por recursos gastos em relação à acurácia e abrangência com as quais usuários atingem objetivos. A satisfação revela-se pela ausência do desconforto e presença de atitudes positivas para com o uso de um produto. A satisfação mede a extensão pela qual os usuários estão livres de desconforto e suas atitudes em relação ao uso do produto. Outras informações adicionais sobre a satisfação do usuário pode ser obtida através do número de comentários positivos e negativos e pela medida da frequência com que os usuários requerem transferência para outro trabalho ou de problemas de saúde relatados ao utilizar o produto. Sendo assim, para

tornar o conhecimento acessível a todas as classes é preciso que este seja disponibilizado em materiais digitais com usabilidade, pois esta qualidade agrega valores como a eficiência, eficácia e satisfação. Considerando-se que o documento digital pode ser utilizado por usuários com distintas necessidades, estabelece o vínculo entre a usabilidade e a acessibilidade de um documento (TORRES e MAZZONI, 2004). Um usuário com deficiência visual utilizando um leitor de tela poderá ter dificuldades ao tentar obter uma informação ao se deparar com uma imagem. A imagem sem uma legenda ou sem um texto alternativo que a descreva, se torna incompreensível para uma pessoa com deficiência visual que utiliza um leitor de tela. Para amenizar essa carência nos documentos digitais recomenda-se a utilização de texto alternativo contendo a mensagem que a imagem quer passar para o usuário (Figura 14.1). Outra alternativa é colocar uma legenda descritiva para cada imagem ou até mesmo a depender do grau de complexidade de uma imagem ou tabela, é cabível colocar uma explicação na parte do texto. Além disso, um documento digital extenso e desestruturado, sem a numeração das páginas e sem a presença de seções e subseções, pode dificultar o seu manuseio, pois será preciso gastar mais tempo para procurar a informação desejada.



Fonte: (TORRES e MAZZONI, 2004).

Figura 14.3. Exemplo de imagem com texto alternativo.

Texto alternativo:

Figura que representa os fatores comuns às práticas de sucesso. Eles se distribuem em um círculo de 6 balões, e dentro de cada balão surge um fator: a compreensão a participação, a parceria, a formação, o empowerment e a formação.

Os autores Torres e Mazzoni(2004), assinalou dentre os princípios de um conteúdo digital com usabilidade e acessibilidade, o controle do usuário. O princípio do controle do usuário, defende que o usuário possa administrar a velocidade que a informação é apresentada, as animações, o tamanho da fonte, entre outros. Delimitando-o ao material digital, o controle do usuário também agrega valores como a escolha do tipo de voz utilizada. Torres e Mazzoni (2004) acrescentam:

> *"Significa permitir que o usuário possa fazer as adaptações a adequadas para a utilização do produto, desde ajustes no conteúdo em exibição, tais como efeitos de ampliação, parar animações, modificar contraste, optar entre o uso monocromático ou policromático etc. Opções quanto à forma de recebimento de arquivos com conteúdos que integram o*

> *documento principal são também desejáveis [...]. Conteúdos digitais associados a cursos que envolvam várias sessões de interação devem guardar as preferências dos usuários, para que eles possam ter esses ajustes à disposição, em suas posteriores interações com o sistema [...]. É sempre desejável que o usuário possa escolher o formato no qual prefere receber e trabalhar com os conteúdos Página de divulgação de do curso."*

14.3 Ambientes de desenvolvimento de DTB

14.3.1 Microsoft Office Word

Fruto de uma parceria entre a *Microsoft* e o consórcio DAISY, o projeto de desenvolvimento com o intuito de melhorar o acesso ao documento eletrônico criou um plug-in gratuito para o Microsoft Office Word. Baseado no padrão internacional para a publicação de conteúdos multimédia acessível (NISO Z39.86, 2012), o plug-in "Save as DAISY[15]" tem o objetivo de tornar acessível à todos os indivíduos o material digital. Esta é uma solução que custa relativamente pouco e simplifica o processo de criação de materiais acessíveis. Com o desenvolvimento desse *plugin* para o *Word* - *software* para edição de texto desenvolvido pela Microsoft - é possível gerar documentos no formato DAISY principalmente com a utilização dos marcadores de estilos. Após a instalação do *plugin* é preciso verificar se este está ativo. Para isso é necessário ir ao menu iniciar, na opção do word, em suplementos e ativar o DAISY *addin*. Uma nova aba será incluída no menu principal do Word, a Accessibility (Figura 14.4). Todas as opções dessa aba estão em inglês.

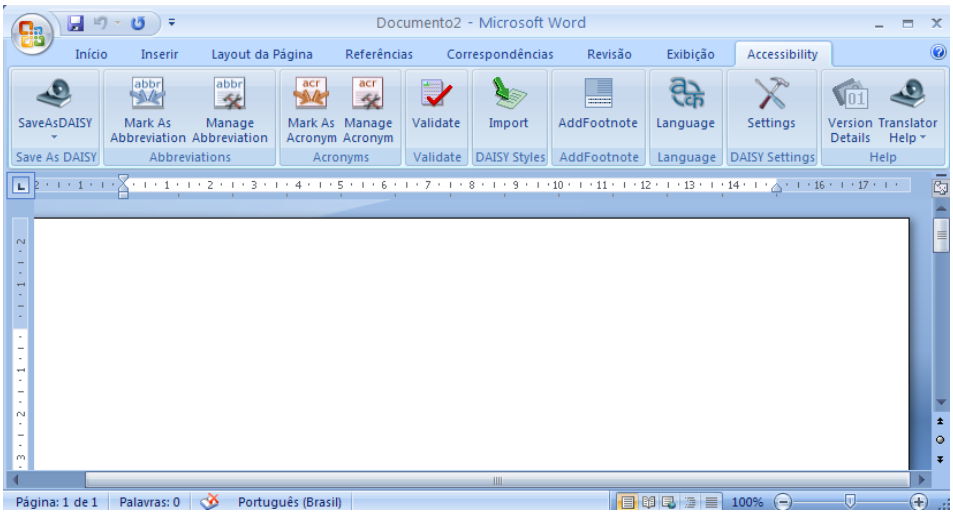


Figura 14.4. Menu de Acessibilidade no *Word*

15 Disponível em: http://www.daisy.org/projects/save-as-daisy-microsoft/

O menu *Accessibility* agrega várias funcionalidades para proporcionar a transformação de um simples documento no *Word* em um arquivo de multimídia, dentre essas funcionalidades, destaca-se a "*Import".* Através do "*Import"* pode-se adicionar os estilos do DAISY aos estilos do *Word.* A marcação por meio dos estilos próprios do DAISY é uma etapa importante para gerar um DTB com qualidade. Há estilos para configuração do autor do documento, títulos, subtítulos, números de páginas, poema, tabela, imagem e outros.

O processo de criação de um DTB no *Word* está dividido em etapas. Primeiro é preciso digitar o texto. Depois é necessário a descrição das imagens utilizando a opção do texto alternativo ou a descrição conjugada no texto. Outro fator importante que está associado a essa segunda etapa é a descrição dos conteúdos não textuais. Podem surgir erros durante a conversão do texto para o formato DAISY relacionados ao tamanho das imagens. Como precaução, recomenda-se ativar a opção para redimensionar as imagens, marcando "*Resize the Image"* no menu "*Accessibility"*, na opção*"*DAISY *Settings",* na caixa "*Image Size Options".* A terceira etapa é a formatação dos principais elementos textuais usando os estilos e os recursos do DAISY. A formatação em níveis dos títulos do texto deve ser com os estilos do Word(Título, Título 1 e etc.). Já para inserção de nota de rodapé, é necessário colocar o conteúdo da nota de rodapé no texto e enumerá-lo. Por exemplo, para adicionar uma nota de rodapé com o significado da palavra CSS, coloca-se a numeração da nota de rodapé e o seu conteúdo (1 Cascading Style Sheets). No texto, após a palavra CSS, coloca-se a mesma numeração (CSS 1). Depois seleciona todo o conteúdo e a numeração para criar a nota de rodapé no menu *Accessibility* na opção "*AddFootnote".* Depois desse processo aparecerá uma janela para o usuário selecionar o contexto em que a nota deve ser inserida e por conseguinte confirmar a configuração (Figura 5.0).

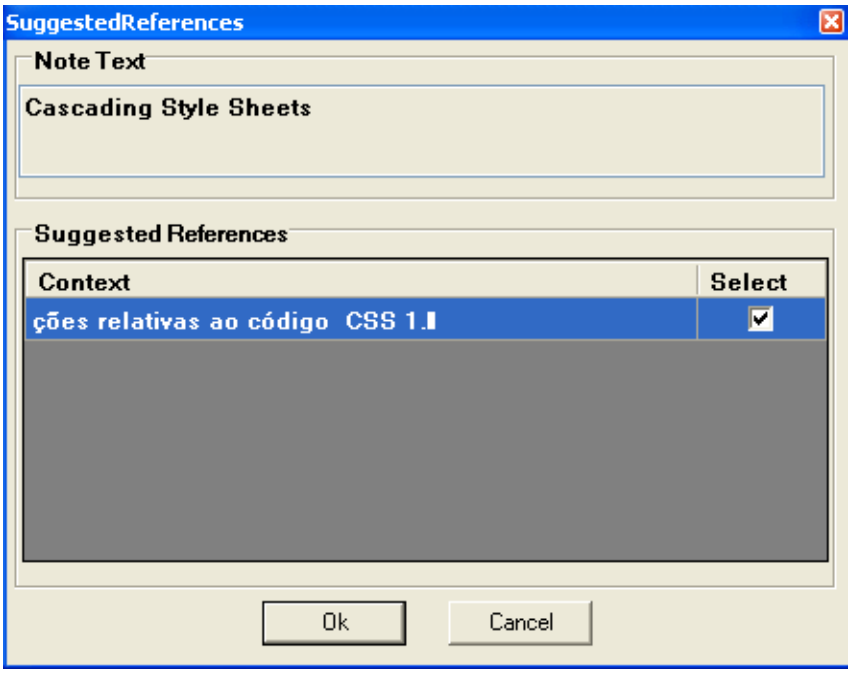


Figura 14.5. Janela de Referências Sugeridas - Nota de rodapé

Há duas maneiras para inserir os números das páginas. O modo automático é o mais simples e confiável, pois só é preciso ativar um recurso no Word. Para isso é preciso marcar como automático a numeração das páginas no menu "*Accessibility",* na opção *"DAISY Settings".* A opção customizada também está disponível e para ser utilizada deve-se inserir os números de cada página manualmente e configurá-los com o estilo "*Page Number DAISY*".

Para configuração de tabelas exitem dois estilos, um que está associado ao título da tabela (Table Caption DAISY) e outro que se encontra no menu "D*esign*", na categoria "Estilos da Tabela", o "*Table-Footer DAISY*". Uma outra funcionalidade agregada a essa ferramenta é a *"Language".* A definição do idioma do texto é importante para um correta leitura pelo leitor de DTB, pois esse tipo de programa emite uma voz sintetizada no idioma do texto. Sendo assim, quando uma parte do texto estiver em outro idioma é preciso selecionar o texto que está em outra língua e na opção *"Language"* indicar o idioma desse texto. Existem duas opções no plugin "*Save as DAISY*" com o objetivo de facilitar a compreensão de acrônimos e abreviaturas. A formatação de um acrônimo ou abreviatura ocorre da seguinte forma: seleciona-se o acrônimo, adiciona seu conteúdo, habilita as configurações para aplicar para todas as ocorrências no texto e para pronunciar o conteúdo do acrônimo pelo leitor de DTB no menu "*Accessibility",* na opção "*Mark As Acronym*". Exemplificando: o acrônimo W3C além de ser descrito na sua primeira utilização em um texto, deverá ter uma descrição configurada no processo mencionado anteriormente, pois durante a execução do DTB quando aparecer esse acrônimo, o seu conteúdo poderá ser lido para o usuário (Figura 6.0). A formatação de abreviaturas ocorre de forma similar. Pórem, não há a opção de pronunciar a abreviatura pelo leitor de DTB.

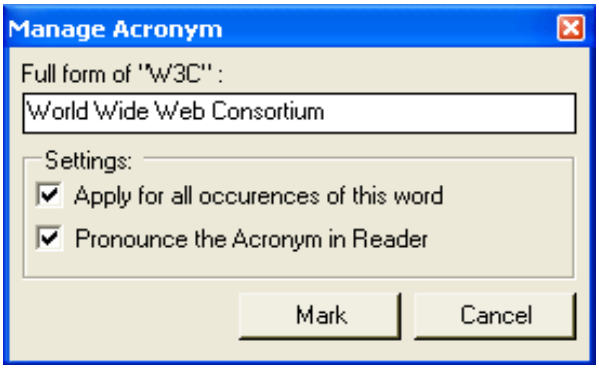


Figura 14.6. Gerenciador de Acrônimos

Os outros estilos do DAISY também são de grande valia. Esses devem ser utilizados de acordo com a necessidade do usuário. Para se destacar uma palavra, sugere-se o estilo "*Word DAISY*". Já para informar o autor do texto, utiliza-se o estilo "*Author DAISY*". A Tabela 14.1 descreve todos os estilos e sua funcionalidade.

Estilos	Definição
Address (DAISY)	Endereço referente ao autor, editora. Pode ser usado para marcar endereço, mas também para telefone, email, ou qualquer informação de localização.
Author (DAISY)	Autor do livro.
Blockquote – Author (DAISY)	Citação em bloco do autor do livro.
Blockquote (DAISY)	Citação em bloco.
Bridgehead (DAISY)	Cabeçalho que não faz parte da hierarquia do livro. Deve ser usado apenas quando os cabeçalhos de estrutura não forem aplicáveis no contexto.
Byline (DAISY)	Autor de um determinado capítulo, ou de um artigo que compõe um livro. Usado quando há mais de um contribuidor na autoria do livro. Pode indicar, além do nome do autor, nome de uma revista de onde foi tirado o artigo, editora, etc.
Citation (DAISY)	Uma referência ou citação para um outro documento.
Code (DAISY)	Fragmento de código de um programa.
Covertitle (DAISY)	Título do livro.
Dateline (DAISY)	Data (ou linha onde está a data) usado quando consta no livro a data da publicação.
Definition (DAISY)	Marca a primeira ocorrência de um termo que será explicado em algum ponto do livro.
Definition Data (DAISY)	Definição de um termo.
Definition Term (DAISY)	Termo a ser definido. Precisa ser seguido pela definição do termo, marcada com o estilo.
Definition Data (DAISY)	Em alguns momentos, não funciona adequadamente.
Div (DAISY)	Subdivisão no livro para marcar uma parte com tratamento de exibição diferente das demais. Exibição incompleta.

Epigraph – Author (DAISY)	Tem exatamente o mesmo comportamento do estilo abaixo, Epigraph (DAISY).
Epigraph (DAISY)	Citação feita no início do livro, por exemplo na página.
Image – Caption (DAISY)	Texto descritivo da imagem. Quando há imagem no livro, deve-se colocar logo após a imagem um texto que a descreva e marcá-lo com este estilo.
Keyboard Input (DAISY)	Entrada pelo teclado. Exibição incompleta.
Line Number (DAISY)	Número da linha. Usado em textos jurídicos, por exemplo, em que é usual mencionar em que linha está um determinado trecho de informação.
List Heading (DAISY)	Cabeçalho de uma lista.
Page Number (DAISY)	Número de página
Poem – Author (DAISY)	Autor do poema. Após este estilo, deve haver uma linha vazia com o estilo “Normal”.
Poem – Byline (DAISY)	Autor de um determinado poema em um livro, ou fonte do poema. Após este estilo, deve haver uma linha vazia com o estilo “Normal”.
Poem – Heading (DAISY)	Cabeçalho em um poema. No entanto, este estilo não funciona adequadamente, o plugin impõe restrições para a geração do livro e normalmente falha na execução do comando Save as DAISY.
Poem – Title (DAISY)	Título de um poema. Após este estilo, deve haver uma linha vazia com o estilo “Normal”.
Poem (DAISY)	Estrofe de um poema. Após este estilo, deve haver uma linha vazia com o estilo “Normal”.
Prodnote - Optional (DAISY)	Nota do produtor, opcional.
Prodnote – Required (DAISY)	Nota do produtor, requerida.
Quotation (DAISY)	Citação
Sample (DAISY)	Exemplo dado pelo autor ao longo do

	livro.
Sent (DAISY)	Sentença. O plugin pode reconhecer as sentenças automaticamente a partir do ponto. Mas se quiser marcar sentenças de forma diferente (por exemplo, as orações separadas por vírgulas) pode usar este estilo. Quando marcar várias sentenças, é importante deixar entre elas um espaço em branco sem marcação. Caso contrário, o plugin irá interpretar que as sentenças são uma só.
Span (DAISY)	Semelhante ao Div, permite destacar uma linha no texto. Exibição incompleta
Table – Caption (DAISY)	Título da tabela. É um texto opcional escrito antes da tabela.
Table – Footer (DAISY)	Rodapé da tabela. Esse texto deverá ser repetido a cada vez que uma linha da tabela for lida. Na verdade, deveria ser o cabeçalho da tabela, mas o plugin do Word o colocou como rodapé. Exibição incompleta
Word (DAISY)	Palavra. Permite marcar as palavras dentro de uma sentença. Exibição incompleta
Título 1, Título 2, Título 3, Título 4, Título 5 e Título 6.	Usados para marcar níveis no livro, por exemplo, capítulos, seções, subseções. Se decidir marcar o primeiro capítulo com o estilo “Título 1”, o segundo capítulo também será marcado com este estilo, bem como todos os capítulos do livro. Use então o “Título 2” para seções, “Título 3” para subseções, e assim sucessivamente. Todo livro precisa ter pelo menos um termo marcado com o estilo “Título 1”. Caso contrário, o plugin irá inserir o termo “Título vazio” no início do livro.

Fonte: (TORRI, 2012)

Tabela 14.1. Estilos do DAISY e suas funcionalidades

Após a edição do texto utilizando os estilos do DAISY é preciso salvar o documento em “.docx”. Por conseguinte, pode-se optar dentre os quatro tipo de DTB

fornecidos pelo Word que estão divididos em duas categorias, para apenas um texto e para múltiplos textos: o "*DAISY XML*" e o "*Full DAISY*" (Figura 7.0). O "*DAISY XML*" da categoria de documentos simples ou apenas um texto converte o documento apenas em XML[16] e não terá um arquivo correspondente em áudio. O "*Full DAISY*" da categoria de documentos simples cria um DTB completo e com aúdio, o mesmo ocorre com o "*Full DAISY*" da categoria de documentos múltiplos, porém este último permite a conversão de vários textos simultaneamente.

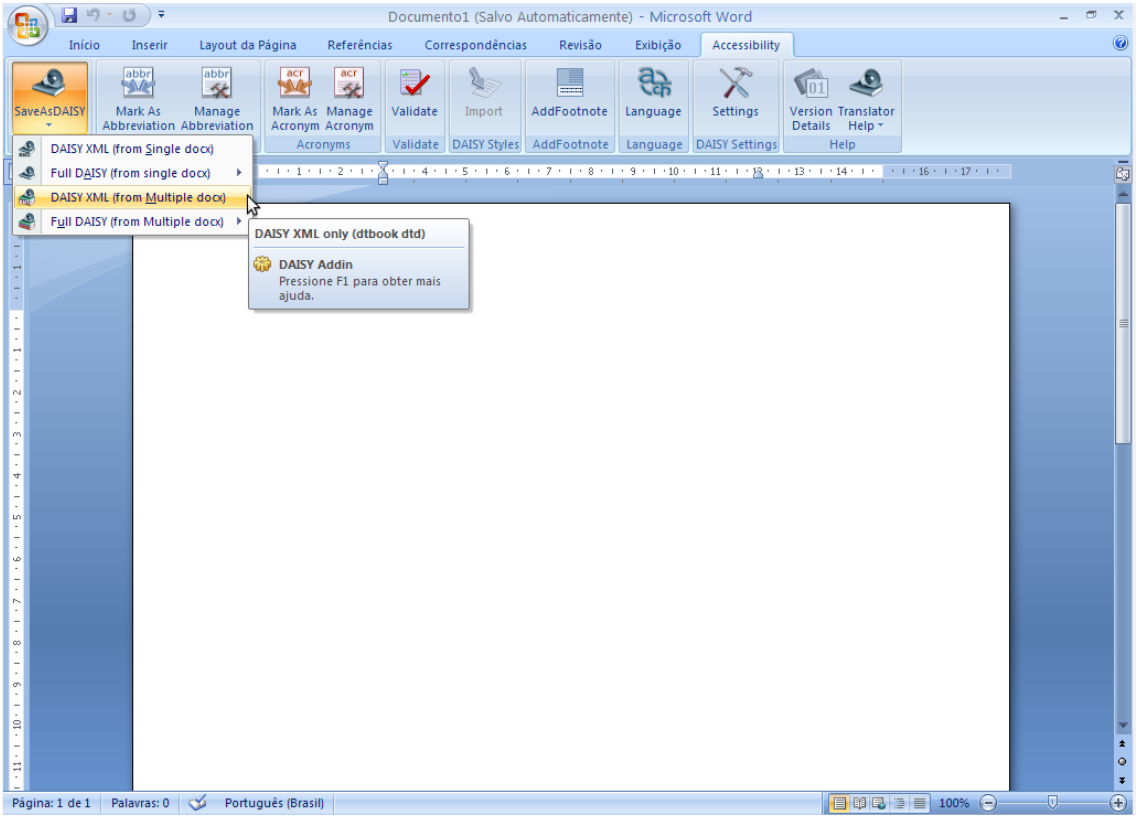


Figura 14.7. Opções de DTBs no Word

O "*DAISY XML*" da categoria de múltiplos documentos traduzirá todos os textos apenas em XML. O tipo de DTB recomendado pelos autores é o "*Full Daisy*" para documentos simples, pois com um DTB desse tipo é possível realizar a leitura visual, anotações, navegabilidade, narração em áudio sincronizada, marcações, apresentação de imagens e formatação para impressão em braille (PARAGUAY, 2005). Depois de escolhido o tipo de livro digital falado a ser gerado e a configuração do documento estiver correta, deve-se informar o diretório de saída, o título do documento, caso ele não tenha sido informado no texto, o autor do documento, a editora e o número de identificação do mesmo (Figura 14.8).

16 XML- Extensible Markup Language

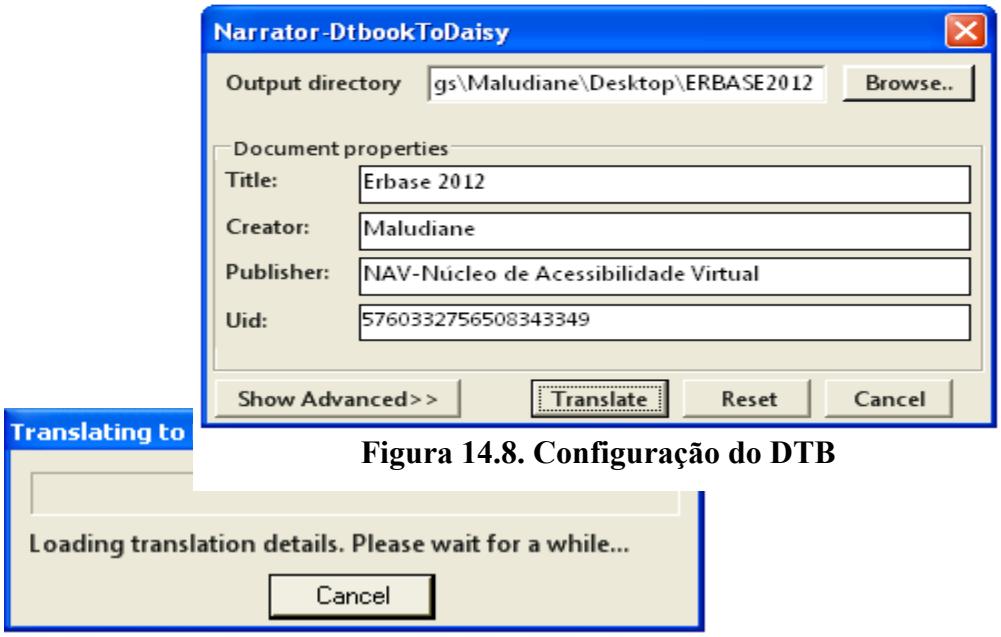


Figura 14.8. Configuração do DTB

Figura 14.9. Primeira etapa da tradução

1.1 As seguintes etapas para converter o documento em um material digital em conformidade com a norma **ANSI / NISO Z39.86**, constituem-se da tradução de documentos Open XML produzidos com o *Microsoft Office Word* em DAISY XML (NISO Z39.86, 2012). A Figura 14.9 exibe a primeira etapa da tradução. Já coma Figura 14.10 pode-se visualizar o progresso da tradução. E a Figura 14.11 mostra o resultado do processo e os possíveis aviso e erros encontrados durante a tradução do documento em um conteúdo de multimédia acessível.

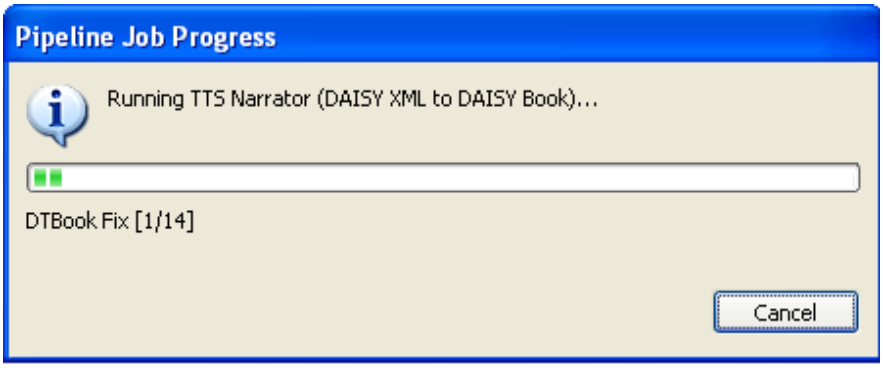


Figura 14.10. Segunda etapa da tradução

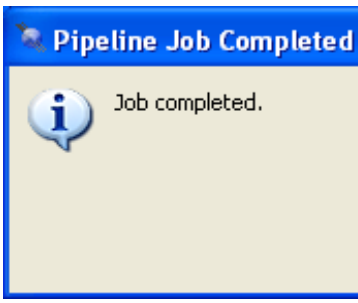


Figura 11.0. Terceira etapa da tradução

Dur ante esse processo, o programa identifica, através da utilização dos estilos DAISY, a estrutura e todo o conteúdo do texto e por conseguinte, criar arquivos semelhantes à arquivos do tipo XML, tais como, o OPF (do inglês, Open Packaging Format), o NCX (do inglês, Navigation Center eXtended) e outros.

14.3.2 Editor Dirce

Dirce é um opção de ambiente para criação de um DTB. O Dirce é um programa gratuito que permite tanto a criação como tocar ou excutar um livro digital falado. É um programa de multiplataforma, ou seja, pode ser executado tanto em sistemas operacionais *Linux* como *Windows.* Sendo assim, o Dirce possui apenas um pré-requisito para funcionar, a máquina virtual *Java.* O editor Dirce possui uma interface simples e acessível, entretanto dispõe apenas de poucos recursos de edição. Para criar um DTB no Dirce é necessário criar um novo projeto, os capítulos do livro, inserir o conteúdo textual ou os arquivos multimídia e depois formatar o texto. Também pode-se criar uma hierárquia com os títulos, subtítulos e assim por diante. A edição do texto que é realizada através de código permite apenas marcar parágrafos, frases, números de páginas, notas e destacar as palavras com negrito (Figura 14.2).

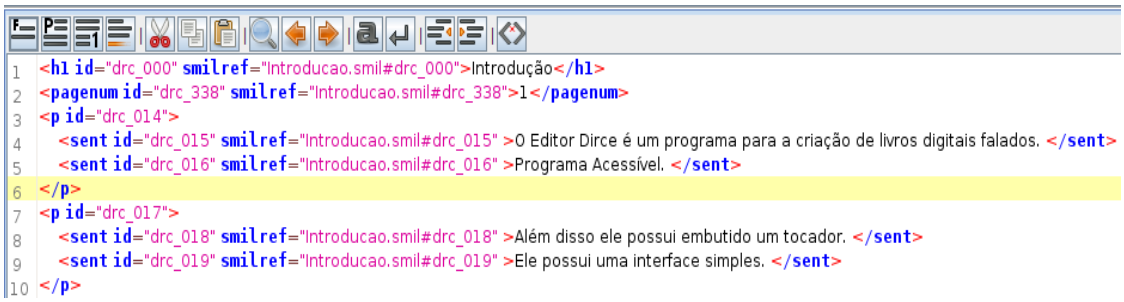


Figura 14.2. Editor Dirce

Conforme ilustrado na Figura 14.2, a edição do documento no Dirce é de maneira distinta do Word. O primeiro capítulo é representado pelo código “h1”. Este código contém o “id” que é a identificação excluisa de cada capítulo e normalmente é informada pelo programa. Já o “smilref” é um código que é usado para indicar o arquivo de multimídia correspondente para a parte do texto que ele está relacionado. Similarmente, os outros códigos para edição do texto no Dirce apresentam uma identificação (id) e uma referência do arquivo de multimídia (smilref). A palavra “smil” vem da linguagem de integração de arquivos de multímidia sincronizados (SMIL) e é o nome dado a extensão dos arquivos de multímidia que comumente associam o texto digitado aos arquivos de áudio nos formatos MPEG (.aac e .mp3) . O “smil” envia os arquivos aos poucos para ser executado, dando a sensação de continuidade. Para ilustar, a leitura de um documento com trinta palavras gera uma gravação de cerca de um minuto. O áudio dessa gravação é dividido a cada dez palavras lidas gerando assim três arquivos de áudio. A associação correta desses arquivos de áudio com a sequência das frases do documento é imprescidível para compreensão da informação apresentada tanto em forma textual quanto em áudio. Em outras palavras, as dez primeiras palavras do documento precisam está sincronizadas com o primeiro arquivo em áudio e não com o último. Para que a sincronização dos arquivos fosse realizada seria necessário a criação de outro arquivo que registrasse ordeiramente a sequência em que os arquivos de áudio e o documento seriam executados. Na criação de DTBs, o arquivo no formato “smill” tem por finalidade indicar o arquivo de áudio correto no momento em que certa parte do texto que está sendo apresentada para o usuário.

Como mencionado anteriormente, qualquer imagem apresentada ao usuário seja em um documento ou em um *site* precisa está descrita corretamente. Por conseguinte, no programa Dirce a inclusão de uma descrição para cada imagem se torna obrigatória, ou seja, a imagem só será inserida no texto caso contenha uma descrição. O dirce também possibilita a inserção de arquivos de áudio (.mp3). Além de gerar a gravação do documento em voz sintetizada, também é permitido colocar, por exemplo, um fundo musical enquanto o documento é lido pelo tocador. Durante o processo de edição no editor Dirce, também é possível reproduzir e visualizar o livro digital falado. Para adaptar as necessidades do usuário, o Dirce agrega funcionalidades tais como: alto contraste, aumentar e diminuir letra, restaurar o padrão de exibição (Figura 14.3).

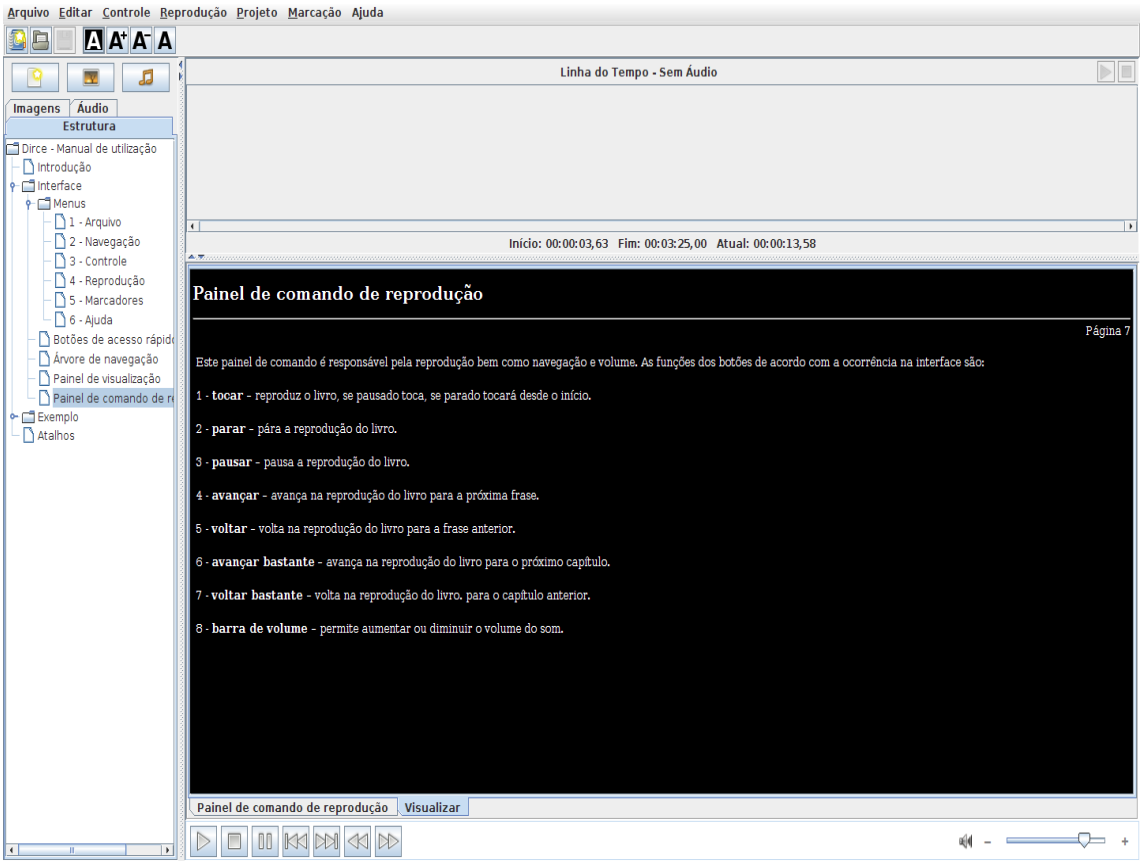


Figura 14.3. Visualização de um DTB no Editor Dirce

Conforme observado na Figura 14.3, o editor Dirce é um programa de computador que atende aos requisitos básicos da acessibilidade virtual. No Dirce o usuário pode adaptar o modo de visualização da informação às suas preferências e pode controlar a velocidade que a informação é reproduzida.

Foram apresentadas duas ferramentas para a produção de material didático acessível, o *Word* e o editor Dirce. Visto que o *Word* é uma ferramenta já conhecida no mercado e comumente utilizada por usuários comuns, ela se torna uma ferramenta mais simples para se criar um DTB. Porém, é válido ressaltar que o *Word* diferentemente do

Dirce não é uma ferramenta gratuita. Não obstante, a criação de um livro digital falado no Dirce é através de códigos e para os usuários não acostumados a trabalhar com códigos pode não ser atraente. De qualquer maneira, essas ferramentas são capazes de gerar materiais digitais acessíveis e com usabilidade.

14.4 Ambientes de execução de DTB

O MECDaisy é uma ferramenta desenvolvida pelo Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro (UFRJ) que também é conhecida como tocador de DTBs (Figura). Advindo de um projeto financiando pelo Ministério da Educação, o Mecdaisy tem o objetivo de permitir a reprodução de livros em formato digital acessível, no padrão DAISY. O MECDaisy áudio, gravado ou sintetizado. Seguindo as diretrizes fixadas pelo padrão DAISy, o MECDaisy apresenta os recursos necessários para navegar pelo texto, reprodução sincronizada de trechos selecionados ou de todo o DTB, o recuo e o avanço de parágrafos e a busca de seções ou capítulos.
Além disso, agrega outras funcionalidades, armazenar anotações no livro, impressão em Braille, controle do tamanho da fonte. Todo texto é indexado, facilitando, assim, o seu manuseio por meio dos níveis e números de páginas (Figura 14.4).

Figura 14.4: MEC DAISY

O MECDaisy executa o livro digital falado, trecho por trecho, permitindo o acompanhamento da leitura principalmente pelas pessoas com deficiência visual. Esse tocador permite o acesso do conteúdo por meio do índice do livro, por níveis e por página. Há também informações sobre os dados gerais do livro, conforme ilustrado na Figura 14.5.

Figura 14.5. Informações sobre o Livro

O DORINA DAISY Reader (DDReader) é outro exemplo de tocador de livro digital falado. Ele é um completo do navegador Mozila FireFox (Figura 14.6). Esse tocador apresenta as mesmas funcionalidades do MECDaisy, armazenar anotações no livro, impressão em Braille, controle do tamanho da fonte, pesquisa por palavras-chave e outros. Porém, a diferença entre o DDReader eo MECDaisy é que não exite, até o momento, o sintetizador de voz gratuito na língua portuguesa. A voz sintetiza padrão do DDReader é o inglês. Para adquirir a voz sintetizada na língua portuguesa é preciso comprá-la. As Figuras 17.0, 18.0, 19.0 a seguir apresentam o DDReader em funcionamento.

Figura 14.7. Histórico do DDReader

14.5 Referências

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 9126/1: Engenharia de software- qualidade de produto: Modelo de qualidade. Rio de Janeiro. 2003.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 9241/11: Requisitos ergonômicos para trabalhos em escritórios com computadores: Orientações sobre usabilidade. Rio de Janeiro. 2002.

BRASIL, Decreto legislativo n.186, de 9 de julho de 2008. Aprova o texto da Convenção sobre os Direitos das Pessoas com
Deficiência e de seu Protocolo Facultativo, assinados em Nova Iorque, em 30 de março de 2007. Artigo 1º. Propósito.Disponível em: <http://vademecumjuridico.blogspot.com/2008/11/decreto-legislativo-com-fora-de-emenda.html>. Acesso em 10 ago. 2011.

DAISY CONSORTIUM. Officially, the ANSI/NISO Z39.86 Specifications for the Digital Talking Book. Disponível em: <http://www.daisy.org/daisy standardwPYO6riKm8xEmLA>. Acesso em 23 jan. 2012.

IBGE. Estatísticas sobre
pessoas com necessidades especiais.
Disponível em: <http://www.ibge.gov.br/home/presidencia/noticias/noticia_visualiza.php?id_noticia=438&id_pagina=1>. Acesso

em: 02 mar. 2012.

__.População com deficiência no mundo e no Brasil. Disponível em: <http://multirio.rio.rj.gov.br/educador/index.php?option=com_k2&view=item&id=322%3Apopula%C3%A7%C3%A3o-com- defici%C3%AAncia-no-mundo-e-no-brasil&Itemid=9>. Acesso em 24 fev. 2012.

GODINHO, Francisco. Tecnologias de Informação sem Barreiras no Local de Trabalho.Disponível em: <http://www.acessibilidade.net/trabalho/Manual%20Digital/capitulo4.htm>. Acesso em 15 jan. 2012.

NISO Z3986. Specifications for the Digital Talking Book. 2005; Disponível em: <http://www.daisy.org/z3986/2005/Z3986-2005.html>. Acesso em 25 jan. 2012
.

PARAGUAY, Ana Isabel B.B; SPELTA, Lêda Lúcia; SIMUFOSA, Miriam Hitomi. DTB (Digital Talking Book), LDF (Livros Digitais Falados), DAISY (Digital Accessible Information SYstem) ou Livros Digitais DAISY – UMA (OUTRA) MANEIRA DE SE LER. III Seminário e II Oficina ATIID. São Paulo, 2005.

TORRES, E. F.; MAZZONI, A. A. Conteúdos digitais multimídia: o foco na usabilidade e acessibilidade**.** Ci. Inf., Brasília, v. 33, n. 2, p. 152-160, Ago. 2004

TORRI, Monique. Tutorial de instalação do MECDaisy e geração do livro digital acessível . Disponível em: <http://*demogimirim.edunet.sp.gov.br/Index/**TutorialMecdaisy.pdf*** >. Acesso em: 09 set. 2011.

Capítulo 15

Internet Embarcada em Microcontroladores PIC

Rodrigo M. Bacurau, Brauliro G. Leal e Felipe P. Correia

Abstract

Since Internet inception, many adaptations have been done in order to make it more versatile and affordable. Nowadays, Embedded Internet has provided the power of connectivity to a variety of embedded devices, such as appliances, handheld devices, industrial machinery, among others. In response to this demand, this course presents an introduction to Embedded Internet technology for Microchip PIC microcontrollers using MPLAB IDE, MPLAB C30 compiler and Proteus ISIS electronic circuits simulator. Step-by-step, we show how to configure Microchip TCP/IP stack. At the end of this course, students will have learnt how to build and simulate an embedded HTTP server using 24FJ128GA006 PIC microcontrollers.

Resumo

Desde sua criação, a Internet vem passando por uma série de transformações que a torna mais versátil e acessível. Atualmente, por meio da tecnologia denominada Internet embarcada, o poder de conectividade a rede mundial de computadores vem sendo estendido a uma variedade praticamente ilimitada de dispositivos embarcados como eletrodomésticos, dispositivos portáteis, máquinas industriais, entre outros. Em resposta a essa demanda, este minicurso apresenta uma introdução à tecnologia de Internet embarcada para os microcontroladores PIC da Microchip, utilizando a IDE MPLAB, o compilador MPLAB C30 e o simulador de circuitos eletrônicos Proteus-ISIS. Nele será demonstrado passo-a-passo como configurar a Pilha TCP/IP da Microchip. Espera-se que ao final do curso os estudantes tenham adquirido conhecimentos básicos para construir e simular um servidor HTTP embarcado utilizando um microcontrolador PIC 24FJ128GA006.

Este trabalho foi publicado nos Anais da XII Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe, em abril de 2012 – UNIVASF -www.erbase2012.univasf.edu.br- ISSN 2177-4692.

15.1. Dados Gerais

15.1.1. Objetivos do Curso

15.1.1.1. Objetivo Geral

Introduzir os principais conceitos acerca da tecnologia Internet embarcada em microcontroladores PIC e motivar a execução de trabalhos futuros usando essa tecnologia.

15.1.1.2. Objetivos Específicos

- Apresentar a definição de internet embarcada;
- Expor exemplos de aplicações desta tecnologia;
- Definir os componentes da Pilha TCP/IP da Microchip e ensinar como usá-la;
- Apresentar e introduzir a utilização de *software*s necessários para a implementação de um sistema embarcado baseado em microcontroladores PIC;
- Construir e simular um servidor HTTP embarcado utilizando um microcontrolador PIC 24FJ128GA006.
- Ensinar a construir um sistema web de monitoramento;
- Ensinar a construir um sistema web de controle.

15.1.2. Tipo de Curso

Este curso será teórico, porém apresentará todos os passos necessários para reprodução futura dos experimentos nele descritos.

15.1.3. Tratamento Dado ao Tema

Apresentação, comparação de tecnologias e formação de novas habilidades.

15.1.4. Perfil Desejado dos Participantes

Nível de graduação, pós-graduação e empresarial. Recomendado o conhecimento prévio de microcontroladores e programação em Linguagem C.

15.1.5. Infraestrutura Necessária

Projetor multimídia e computador.

15.2. Conteúdo do Minicurso

Este mini-curso seguirá o seguinte roteiro:

- Introdução à Internet Embarcada: nesta seção serão discutidos os principais conceitos a respeito da Internet embarcada, suas vantagens em relação a tecnologias semelhantes, e apresentadas algumas aplicações para essa tecnologia: automação industrial, VOIP, IPTV e domótica.

- Componentes Necessários: nesta seção serão apresentados os principais componentes constituintes de um sistema de internet embarcada: pilha TCP/IP, controlador ethernet, interface de rede e microcontrolador.
- Introdução à Pilha TCP/IP da Microchip: nesta seção serão apresentadas as características e funcionalidades da Pilha TCP/IP da Microchip.
- *Software*s Úteis: nesta seção serão apresentados os *software*s necessários para desenvolver um projeto utilizando a solução de internet embarcada da Microchip, como instalá-los e configurá-los. Serão utilizados os seguintes *software*s:
    - MPLAB IDE (v. 8.56): Ambiente de desenvolvimento integrado para microcontroladores PIC;
    - MPLAB C30 (v 3.24 Lite): Compilador para microcontroladores PIC24;
    - PILHA TCP/IP MICROCHIP (v. 5.25): API fornecida pela Microchip para o desenvolvimento de aplicações de Internet Embarcada;
    - PROTEUS ISIS (v. 7.6): Simulador de sistemas microcontrolados;
    - WinPcap (v. 4.1.2): programa para análise de rede e captura de pacotes para plataforma Windows.
- Configuração da Pilha TCP/IP: nesta seção será apresentada a estrutura de diretórios da Pilha TCP/IP da Microchip, exemplos de projetos utilizando essa API, exemplos de simulação no Proteus-ISIS de placas de desenvolvimento da Microchip, a funcionalidade de cada arquivo da Pilha TCP/IP Microchip e como configurá-los. A partir de um projeto base será desenvolvido um projeto simples utilizando cooperativismo multitarefa.
- Monitoramento Web: nesta seção será desenvolvido um sistema web para monitoramento do estado de LEDs, botões, chaves e IP de um circuito eletrônico conectado a uma rede TCP/IP.
- Controle Web: nesta seção, o sistema construído anteriormente será incrementado, permitindo controlar, via web, o acionamento dos LEDs e o envio de mensagens de texto para um display de um circuito eletrônico conectado via rede TCP/IP.

15.2.1. Introdução à Internet Embarcada

Várias redes de comunicação, como a TV, o rádio e até mesmo algumas tecnologias usadas na automação industrial são analógicas. Porém, devido às inúmeras vantagens da tecnologia digital, a tendência é que todas essas redes migrem para esta tecnologia. As redes digitais possuem inúmeras vantagens quando comparadas com as analógicas, tais como: suportam maior tráfego de dados, necessitam de menos cabos e são mais robustas (BALTAZAR, 2008). As vantagens de se usar comunicação digital se ampliam quando é utilizado o protocolo de comunicação TCP/IP para a comunicação de dispositivos embarcados. Tal tecnologia é denominada Internet embarcada. A Internet embarcada é uma tecnologia que permite a conexão de sistemas embarcados à Internet, especialmente sistemas microcontrolados e constituídos por DSPs (*Digital Signal Processors*) (SANTOS, 2009). Na Figura 15.22 é mostrada algumas placas que utilizam a tecnologia de Internet embarcada.

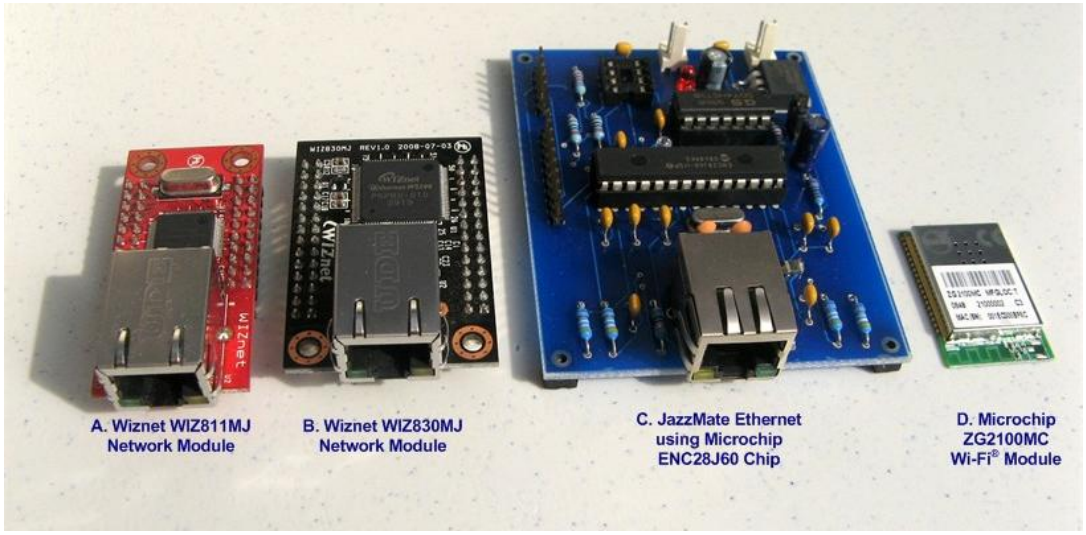


Figura 15.22. Exemplos de placas que utilizam a tecnologia de Internet embarcada.

A tecnologia de Internet embarcada possui as seguintes vantagens com relação às demais tecnologias de comunicação digital:

- Permite a utilização dos mesmos equipamentos utilizados para compor as redes de computadores, como switches e roteadores. Esses equipamentos já estão no mercado há bastante tempo, possuem tecnologia bem consolidada e preço reduzido quando comparado a tecnologias similares;
- Permite a utilização da rede mundial de computadores, a Internet, para controle a distância, dispensando a criação de redes privadas;
- É uma tecnologia bastante consolidada. Existe uma série de protocolos bem definidos para as mais diversas necessidades de comunicação utilizando essa tecnologia;
- Possui boa escalabilidade, ou seja, permite a expansão da rede com a conexão de novos nós sem que seja necessário mudar a estrutura da rede existente;
- Possui a característica de interoperabilidade, ou seja, equipamentos de diversos fabricantes comunicam-se entre si, podendo ser usados para compor uma rede. Essa característica é facilitada pela definição bem definida dos protocolos TCP/IP.

A tecnologia de Internet embarcada é muito utilizada para possibilitar o monitoramento e acionamento remoto. Para tal, utiliza-se um *hardware* embarcado conectado à rede, o qual oferece uma interface HTML, por exemplo, e permite tanto a aquisição de dados do sistema embarcado quanto o envio de comandos. Também é usada para automação industrial, IPTV, VOIP, domótica e inúmeras outras aplicações.

Antigamente, para fazer monitoramento ou acionamento remoto, utilizava-se um computador e uma placa de aquisição de dados. Esse modelo pode ser visto na Figura 15.23. Através do mesmo é possível realizar acionamento/monitoramento, tanto em uma rede local quanto na Internet.

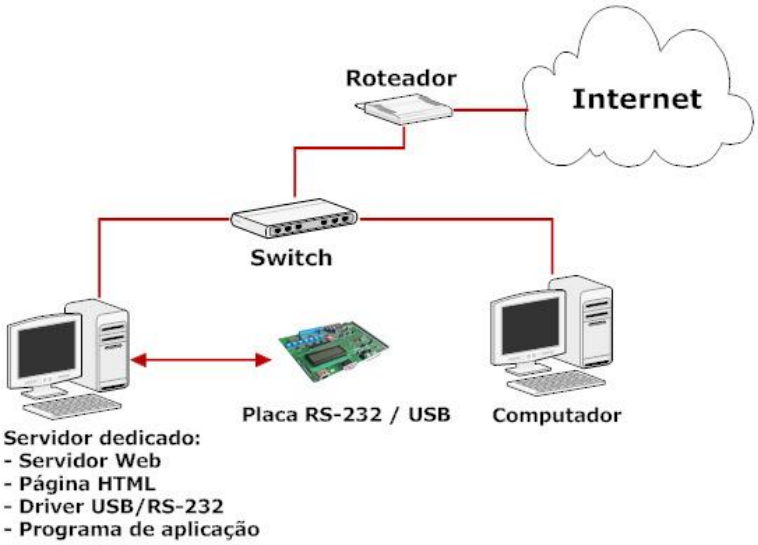


Figura 15.23. Infraestrutura tradicional para um sistema de acionamento ou monitoramento via web (SANTOS, 2009).

No modelo tradicional é necessário um computador para prover a interface de comunicação com a rede TCP/IP. O computador é conectado na rede através de Wi-Fi ou de um conetor RJ-45e comunica-se com o sistema embarcado através de uma conexão RS-232 ou USB, por exemplo. No computador é configurado um servidor web utilizado para publicar páginas HTML na Internet que fornecerão a interface para o usuário interagir com a placa.

Quando se utiliza Internet embarcada para fazer monitoramento ou acionamento, o modelo é modificado e o servidor dedicado não é mais necessário. O próprio circuito embarcado contém todos os componentes necessários para comunicação via rede TCP/IP. Esse modelo pode ser visto na Figura 15.24.

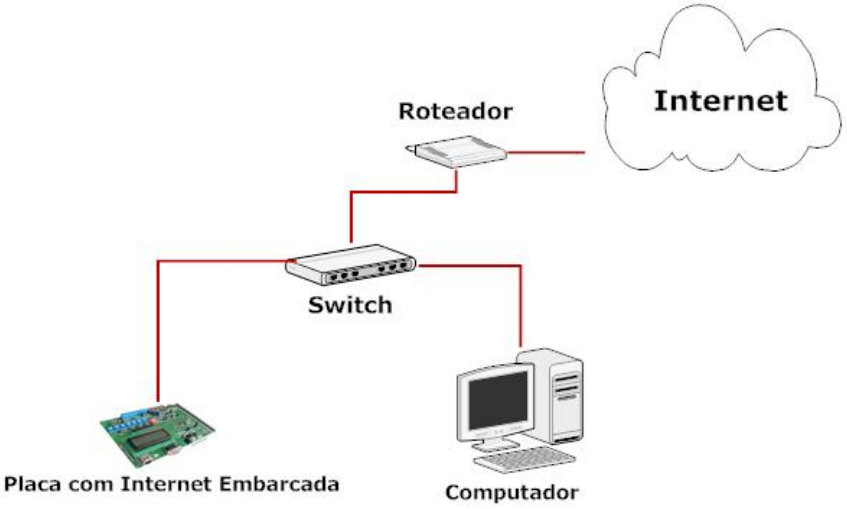


Figura 15.24. Infraestrutura de Internet Embarcada para um sistema de acionamento ou monitoramento via web (SANTOS, 2009).

No modelo da Figura 15.24, as páginas HTML do sistema estão armazenados na própria placa onde está implementado um servidor web. Esse modelo possibilita que

monitoramento e acionamento possam ser realizados a partir de uma rede local ou da Internet, utilizando um dispositivo dedicado e de baixo custo.

15.2.2. Componentes Necessários

Para que um sistema embarcado possa conectar-se à Internet, o mesmo deve possuir os seguintes componentes (SANTOS, 2009):

- Interface de rede: necessária para realizar a conexão física do sistema embarcado com a rede. O padrão mais utilizado é o RJ-45;
- Controlador Ethernet: responsável por codificar, no padrão Ethernet, as informações recebidas/enviados do microcontrolador ou DSP;
- Implementação do protocolo TCP/IP: *software* embarcado no microcontrolador que implementa os protocolos de comunicação, a fim de estabelecer a conexão lógica com uma máquina remota em um determinado segmento de rede.

Os principais componentes físicos de um sistema de Internet embarcada estão apresentados na Figura 15.25.

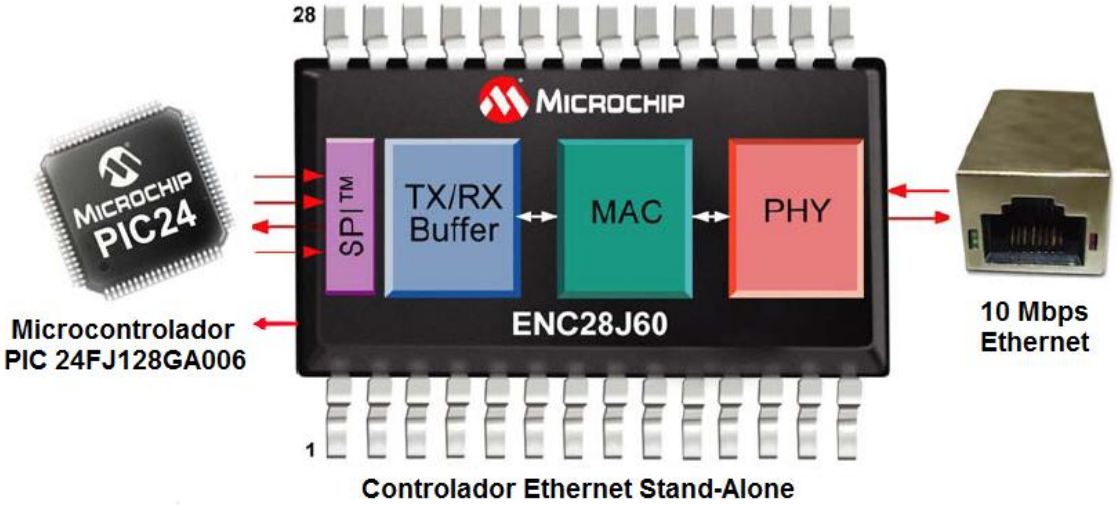


Figura 15.25. Principais Componentes Físicos de um sistema de Internet embarcada.

15.2.2.1. Interface de Rede

Para prover a comunicação com o ambiente externo ao sistema embarcado é necessária uma interface de rede. Deve ser usada uma interface de rede 100/10 Base-TX RJ45 com transformador isolador interno e compatível com o padrão 802.3 (Ethernet). A interface de rede é similar à vista na Figura 15.26.

Figura 15.26. Imagem de uma interface de rede 100/10 Base-Tx RJ45.

15.2.2.2. Controladores Ethernet

Um controlador Ethernet tem a função de receber uma informação e gerar como resultado um dado equivalente codificado no padrão IEEE 802.3 (Ethernet) (SANTOS, 2009).

Neste minicurso será utilizado o controlador Ethernet da Microchip Technology ENC28J60 (MICROCHIP, 2008a). O ENC28J60, Figura 15.27, é um controlador Ethernet *stand-alone*, com o padrão industrial SPI para comunicação serial com outros dispositivos.

Figura 15.27. Chip ENC28J60 em encapsulamento DIP.

O ENC28J60 reúne todas as especificações IEEE 802.3 e incorpora um conjunto de esquemas de filtros de pacote para limitar a quantidade de pacotes que realmente necessitam ser processados. Além disso, o mesmo possui um módulo de DMA *(Direct Memory Access*) dedicado, para permitir uma maior vazão de dados e uma unidade de *hardware* dedicada para cálculo do *checksum*, que é uma informação para detecção de erro utilizada em diversos protocolos de rede. A comunicação com um controlador *host* (e.g. um microcontrolador) é implementada via pino de interrupção e interface SPI a uma velocidade superior a 20 MHz. Dois LEDs são utilizados para indicar a atividade de rede (MICROCHIP, 2008a).

O diagrama de blocos simplificado desse controlador Ethernet pode ser visto na Figura 15.28.

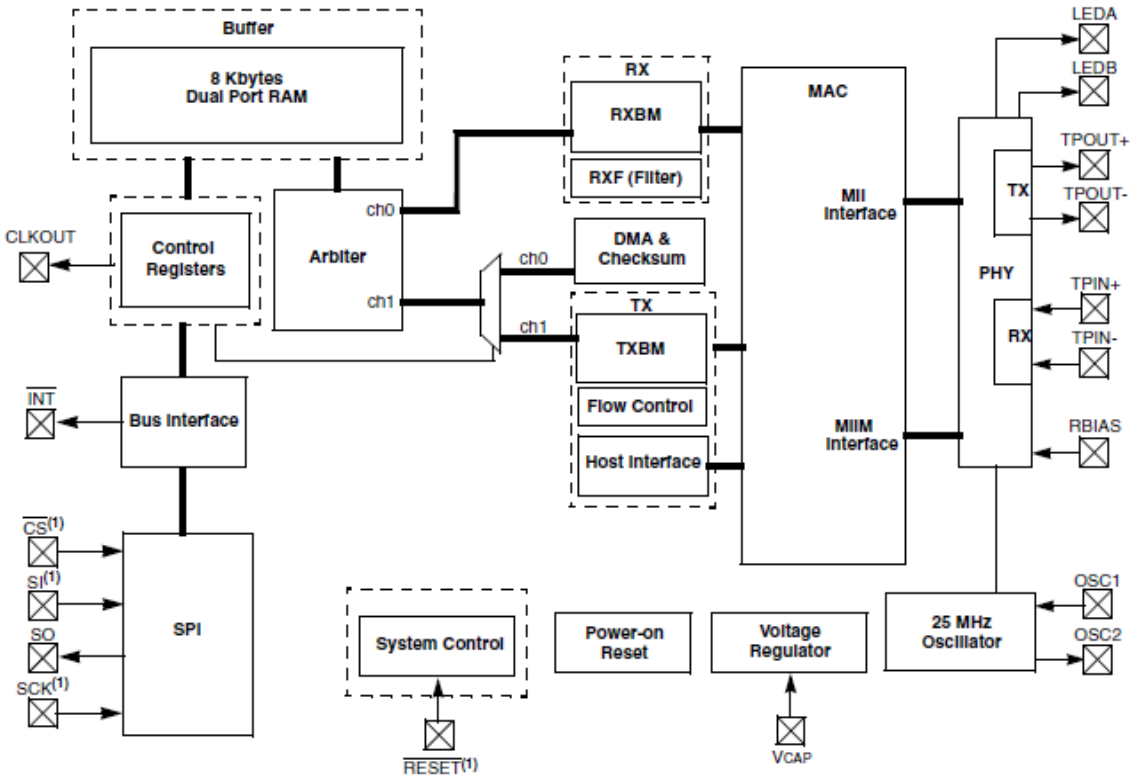


Figura 15.28. Diagrama de blocos simplificado do controlador Ethernet ENC28J60 (MICROCHIP, 2008a).

O ENC28J60 é formado por sete blocos funcionais (MICROCHIP, 2008a):

- Uma interface SPI que permite a comunicação do controlador *host* com o ENC28J60;
- Registradores de controle que são utilizados para controle e monitoramento do ENC28J60;
- *Dual Port RAM buffer* para transmissão e recepção de pacotes de dados;
- Árbitro para controlar o acesso ao buffer RAM quando requisições são feitas simultaneamente pela DMA e pelos blocos de transmissão e recepção;
- Uma interface de barramento que interpreta dados e comandos recebidos via interface SPI;
- Módulo, ou camada, MAC (*Medium Access Controller*) que implementa o padrão IEEE 802.3;
- Camada física (PHY) que codifica e decodifica o dado analógico presente na interface de par trançado.

Esse controlador Ethernet também suporta outros blocos, como oscilador, regulador de tensão *on-chip*, transformadores de nível de tensão para prover tolerância de E/S de 5 V, entre outros (MICROCHIP, 2008a).

Um circuito genérico utilizando o ENC28J60 pode ser visto na Figura 15.29. Além do controlador, um transformador e alguns componentes passivos são necessários para conectar um microcontrolador a uma rede Ethernet.

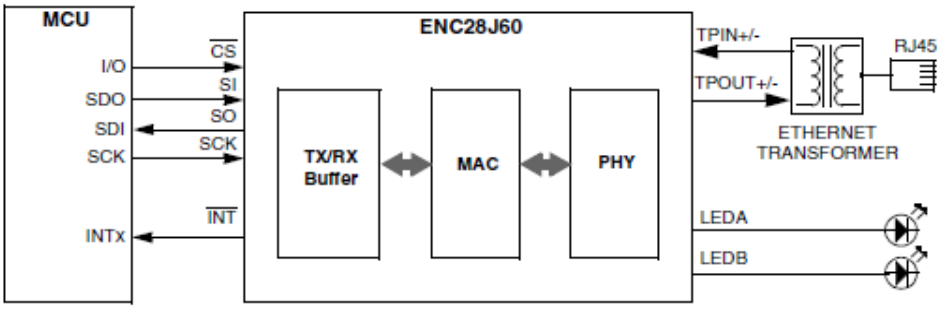


Figura 15.29. Diagrama do circuito de aplicação do ENC28J60 (MICROCHIP, 2008a).

Para estabelecer uma interface Ethernet de fato, o ENC28J60 necessita de um conjunto de componentes padronizados, que devem ser instalados externamente. Esses componentes devem ser conectados como na Figura 15.30.

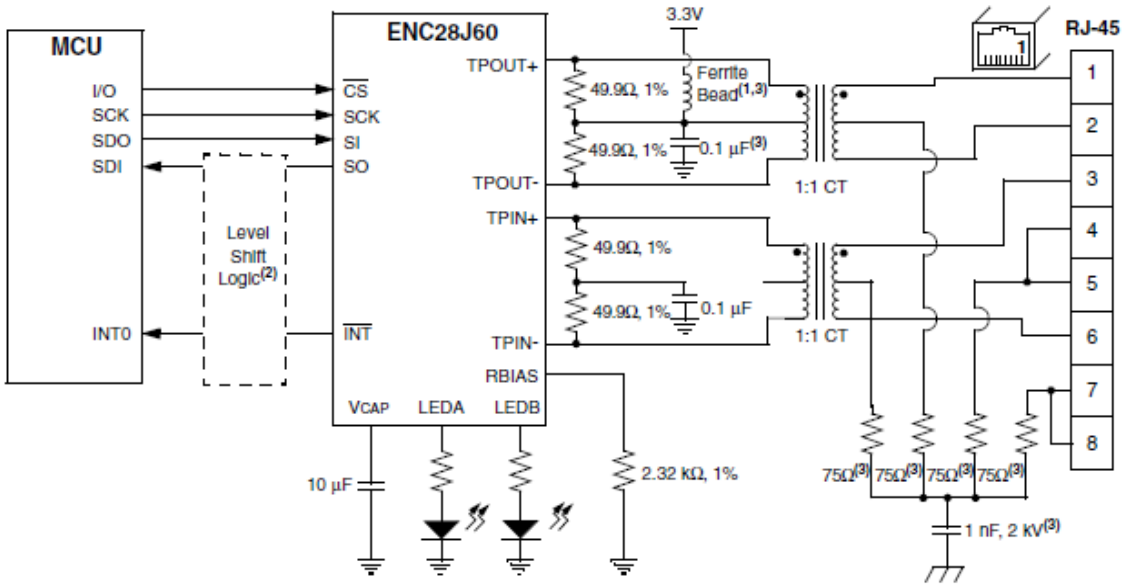


Figura 15.30. Componentes padronizados necessários para estabelecer uma interface Ethernet (MICROCHIP 2008a).

Os circuitos analógicos internos do módulo PHY necessitam de um resistor externo de 2,32 kΩ ± 1% anexado do RBIAS à referência de terra. Esse resistor influencia na amplitude dos sinais TPOUT+ / TPOUT-. Dessa forma, esse resistor deve ser colocado o mais próximo possível do controlador, mas afastado das demais trilhas da placa de circuito impresso, a fim de evitar ruído capacitivo que pode afetar o sinal de transmissão. Além disso, alguns dispositivos (ou componentes) de lógica digital internos ao controlador operam na tensão de 2,5 V. O próprio controlador possui um regulador de tensão interno para gerar essa voltagem. O único componente externo necessário é um filtro capacitivo, conectado do Vcap à referência de terra. Um transformador de pulso 1:1 com *tap* central é necessário nos pinos TPIN+/TPIN- e TPOUT+/TPOUT- para a operação Ethernet.

15.2.2.3. Pilha TCP/IP da Microchip

Existem inúmeras implementações comerciais e não comerciais do protocolo TCP/IP para microcontroladores PIC. A Microchip fornece gratuitamente uma implementação, denominada Pilha TCP/IP Microchip. A Pilha TCP/IP da Microchip consiste em um conjunto de programas que provêm serviços para as aplicações padrões baseadas no protocolo TCP/IP, como: servidor HTTP, cliente email SMTP, protocolo SNMP, Telnet, TFTP entre outros (MICROCHIP, 2008b).

As principais características da Pilha TCP/IP da Microchip são:

- É constituída de forma modular;
- Escrita em linguagem C;
- Compatível com os microcontroladores PIC18 (MPLAB C18 e Hi-Tech PICC-18), PIC24 (MPLAB C30) e PIC32(MPLAB C32);
- Independente de Sistemas Operacionais (SOs);
- Suporte a Sockets TCP e UDP.

A pilha TCP/IP da Microchip é similar a uma pilha TCP/IP tradicional usada nos microcomputadores, porém não implementa todos os módulos. Ela foi desenvolvida levando em consideração todas as especificações contidas nas RFCs de cada um dos protocolos que ela implementa, já que o objetivo da mesma é permitir a conexão de microcontroladores à Internet de forma transparente, como se fosse um *host* qualquer, ou seja, o fato de ser um dispositivo microcontrolado e não um computador, por exemplo, não é percebido.

A maioria das implementações TCP/IP seguem o Modelo de Referência TCP/IP. Este modelo de referência implementa os protocolos em camadas, as quais são "empilhadas" umas sobre as outras (daí o nome pilha TCP/IP). Cada camada acessa os serviços de uma ou mais camadas diretamente abaixo (MICROCHIP, 2008b). Na Figura 15.31 é ilustrado um modelo simplificado da pilha TCP/IP, o qual é implementado pela Microchip. Alguns serviços implementados pelas versões mais recentes da Pilha TCP/IP da Microchip não estão apresentados nesta figura, principalmente os serviços da camada de aplicação.

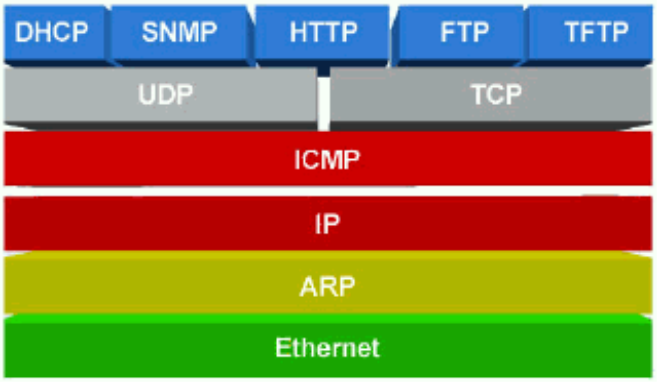


Figura 15.31. Organização da pilha TCP/IP da Microchip (MICROCHIP, 2008b).

A Pilha TCP/IP da Microchip é dividida em várias camadas escritas em linguagem de programação C para os compiladores Microchip HI-TECH, PICC-18, C18, C30 e C32.

A implementação de cada uma das camadas é feita em arquivos separados, enquanto que os serviços e a API são definidas através de arquivos “header/include” (MICROCHIP, 2008b).

A pilha TCP/IP da Microchip é chamada de “pilha viva”, isto é, algumas camadas podem executar operações de forma assíncrona. Para permitir essa característica e ainda estar relativamente independente da aplicação principal que utiliza seus serviços, a pilha TCP/IP da Microchip utiliza uma técnica largamente conhecida, chamada de cooperativismo multitarefa. Em um sistema cooperativo multitarefa as tarefas são executadas sequencialmente e de modo independente. O programa principal envia cada tarefa para ser processada e, ao final da sua execução, o controle retorna ao programa principal e só então a próxima tarefa fica apta a ser executada (MICROCHIP, 2008b).

Geralmente o cooperativismo multitarefa é implementado pelo sistema operacional ou pela própria aplicação principal. A pilha TCP/IP da Microchip foi desenvolvida para ser independente de qualquer sistema operacional e, por isso, ela implementa seu próprio mecanismo cooperativo multitarefa. Dessa forma, ela pode ser utilizada em qualquer sistema, independente se o mesmo utiliza o sistema cooperativo multitarefa ou não. De qualquer forma, uma aplicação que utiliza a pilha TCP/IP da Microchip deve também utilizar o método cooperativo multitarefa. Isso é feito através da divisão de um programa em múltiplas tarefas, ou organizando o programa principal como uma máquina de estados finita (FSM) e dividindo um programa grande em diversos programas pequenos (MICROCHIP, 2008b).

15.2.2.3.1. Servidor HTTP da Microchip

O servidor HTTP2 incluído na pilha TCP/IP da Microchip é implementando como uma tarefa cooperativa que coexiste com a pilha TCP/IP e a aplicação do usuário. A implementação desse servidor é feita no arquivo “HTTP2.c”, com a aplicação do usuário sendo responsável pela implementação de duas funções do tipo *callback*, ou seja, funções que são chamadas automaticamente pelo servidor HTTP e que devem executar alguma tarefa a ser definida pelo usuário (MICROCHIP, 2008b).

Esse servidor não implementa todas as funcionalidades HTTP. Na verdade, o servidor HTTP é uma implementação mínima para sistemas embarcados, que leva em consideração as características e limitações dos mesmos. Mas, de acordo com a Microchip, o usuário pode realizar implementações de outras funcionalidades e/ou recursos definidos pela RFC do protocolo.

Algumas funcionalidades do servidor HTTP podem ser vistas a seguir (MICROCHIP, 2008b):

- Suporte a múltiplas conexões HTTP;
- Dispõe de um sistema de arquivos próprio (MPFS2);
- Suporta páginas web localizadas na memória de programa do microcontrolador ou em memória EEPROM serial externa;

- Suporta os métodos de envio GET e POST (outros métodos podem ser adicionados);
- Suporta um tipo de CGI (Common Gateway Interface) para executar funcionalidades pré-definidas a partir de um navegador remoto;
- Suporta geração dinâmica de conteúdo de páginas web.

15.2.2.3.2. Microchip File System 2 (MPFS2)

O servidor HTTP da Microchip utiliza um sistema de arquivos simples, o MPFS2 (Microchip File System 2) para armazenar as páginas web no microcontrolador. A imagem do sistema de arquivos MPFS2 pode ser armazenada na memória de programa do microcontrolador ou em uma memória externa serial do tipo EEPROM. O MPFS2 segue um formato especial para armazenar múltiplos arquivos em uma unidade de armazenamento (MICROCHIP, 2008b). Esse formato pode ser visto na Figura 15.32.

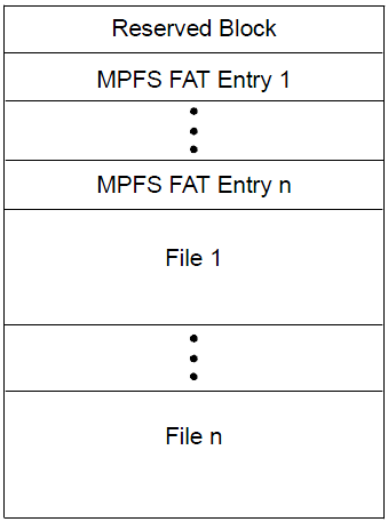


Figura 15.32. Formato de armazenamento utilizado pelo MPFS2 (MICROCHIP, 2008b).

O tamanho do bloco denominado "*Reserved Block*" é definido pela macro MPFS_RESERVE_BLOCK, que pode ter seu valor alterado, se o usuário assim desejar. Esse bloco reservado pode ser utilizado pela aplicação principal para armazenar dados de configuração. O MPFS2 inicia com uma ou mais entradas MPFS FAT (*File Allocation Table*), seguida de um ou mais arquivos de dados. A entrada FAT descreve o nome do arquivo, localização e seu estado. O formato da entrada FAT é apresentado na Figura 15.33.

Flag	Address	File Name
(8 bits)	(16 or 24 bits)	(8-byte + 3-byte format)

Figura 15.33. Formato da entrada FAT utilizada pelo MPFS2 (MICROCHIP, 2008b).

O byte de *flag* indica se a entrada atual está sendo utilizada, foi apagada ou está no fim da FAT. Cada entrada FAT contém 16 ou 24 bits de endereçamento. O tamanho do endereçamento é determinado pelo tipo de memória utilizada, bem como o modelo de tamanho de memória (*large* ou *small* com 16 e 24 bits respectivamente). Sempre que uma memória serial externa EEPROM é utilizada, o endereçamento de 16 bits é utilizado, independente do modelo de tamanho de memória utilizado. Isso implica em uma imagem do MPFS2 de no máximo 64 Kb para a memória externa (MICROCHIP, 2008b).

O campo de endereço em cada entrada FAT contém uma referência para um bloco de dados que contém os dados do arquivo em questão. O formato do bloco de dados pode ser visto na Figura 15.34. O bloco é terminado com um bit de *flag* especial de 8 bits chamado EOF (*End Of File*), seguido do valor FFFF (para endereçamento de 16 bits) ou FFFFFF (para endereçamento de 24 bits). Se a parte de dados do bloco contém um caractere EOF, o mesmo é preenchido por um caractere de "escape".

Data	EOF	FFFFh or FFFFFFh
(variable length)	(8 bits)	(16 or 24 bits)

Figura 15.34. Formato do bloco de dados do MPFS2 (MICROCHIP, 2008b).

O MPFS2 utiliza nomes pequenos para os arquivos: 8 bits para o nome dos arquivos e 3 bits para a extensão, por exemplo: NNNNNNNN.EEE. Um endereço de 16 bits indica o início do primeiro arquivo do bloco de dados. Todos os nomes de arquivo são armazenados em letra maiúscula para facilitar o processo de comparação dos nomes.

15.2.3. *Softwares* Úteis

Nesta seção são apresentadas de forma breve as principais ferramentas utilizadas neste minicurso.

15.2.3.1. MPLAB IDE

O MPLAB é um IDE (Ambiente de Desenvolvimento Integrado) gratuito desenvolvido pela Microchip para seus microcontroladores PIC. O MPLAB é disponível somente para plataforma Windows 32 bits. Ele dispõe de um conjunto de ferramentas (editor, simulador, gravador, depurador e compilador) para o desenvolvimento de aplicações embarcadas, baseadas nos microcontroladores da linha PIC e DsPIC da Microchip. O MPLAB é de fácil utilização e possui um conjunto de aplicativos para desenvolvimento e depuração de forma rápida e ágil de aplicações embarcadas baseadas nos produtos da Microchip (MICROCHIP, 2008c). Uma tela do MPLAB é apresentada na Figura 15.35. Neste minicurso será usada a versão 8.56 desse *software*.

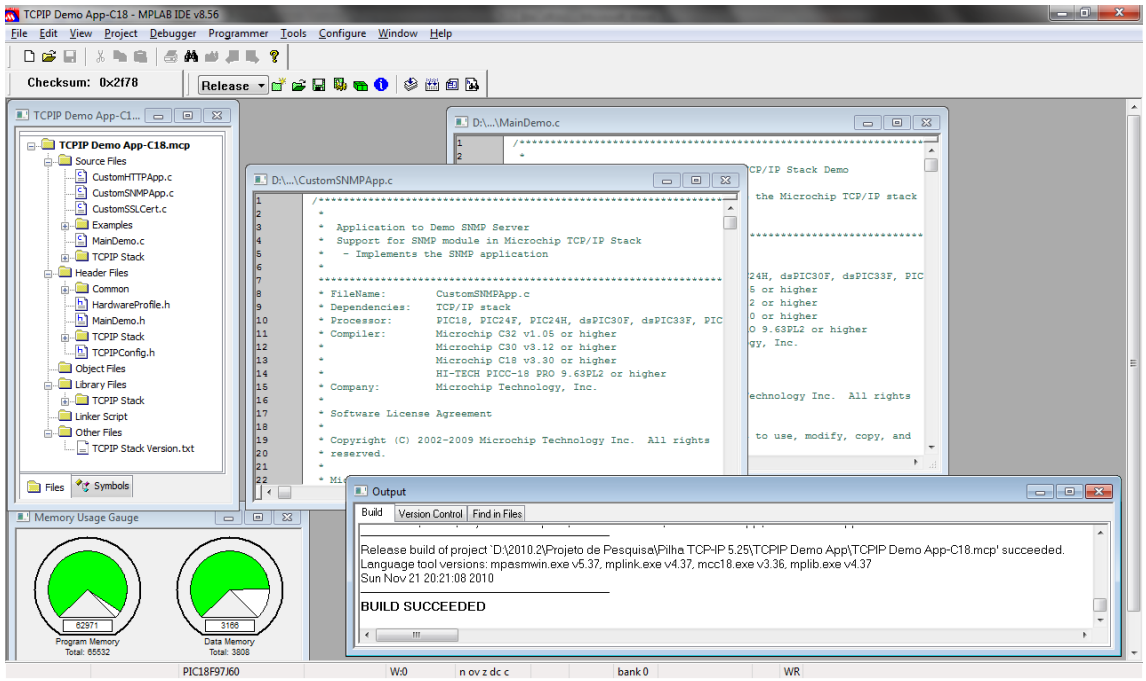


Figura 15.35. Ambiente de trabalho do MPLAB IDE.

15.2.3.2. Compilador MPLAB C30 para microcontroladores PIC24 MCUs e dsPIC DSCs

O MPLAB C30 é um compilador comercial completo para a linguagem de programação "C" ANSI, desenvolvido para as linhas de microcontroladores PIC24, dsPIC30 e dsPIC32 da Microchip. O MPLAB C30 executa como aplicativo console de 32 bits no sistema operacional Microsoft Windows e integra-se com o MPLAB IDE, permitindo depuração em nível de código com o *software* MPLAB e em nível de *hardware* com depuradores como o ICD2BR.

A organização do projeto, opções de compilação e customização de arquivos *linker* pode ser toda feita pelo MPLAB IDE, que disponibiliza uma interface gráfica amigável para o compilador.

As principais características do compilador MPLAB C30 são:

- Compatibilidade com C ANSI '89;
- Integração com o MPLAB IDE a fim de facilitar a gerência de projeto e a depuração em nível de código-fonte;
- Compatibilidade com o *assembler* MPASM, o que permite total liberdade para integração de código C e Assembly;
- Eficiente geração de código com múltiplos níveis de otimização;
- Disponibilidade de uma versão gratuita para estudantes.

Neste minicurso será usada a versão 3.24 Lite desse *software*.

15.2.3.3. PROTEUS VSM

O Proteus é um conjunto de programas para o desenvolvimento de sistemas eletrônicos. Ele possui ferramentas para projeto de placas de circuito impresso (ARES), para criação de protótipos de circuitos (ISIS) e simulação animada de circuitos (VSM). O Proteus permite co-simulação de sistemas microcontrolados, tanto em alto nível quanto em baixo nível integrado com o simulador de circuitos SPICE.

A princípio, é possível desenvolver e testar um projeto utilizando o Proteus VSM antes mesmo de o protótipo físico estar construído. Isso é possível porque o Proteus VSM simula, através de animação em tempo real (ou bem próximo disso), a integração e funcionamento de todos os componentes de um projeto e oferece instrumentos virtuais (Osciloscópio, Multímetro, Analisador lógico, dentre outros) para depuração dos circuitos.

Na Figura 15.36 pode ser vista a placa PICDEM2+, desenvolvida pela Microchip, para treinamento e desenvolvimento de sistemas baseados nos microcontroladores da família PIC 18F. Na Figura 15.37 pode ser vista a mesma placa simulada no Proteus VSM.

Figura 15.36. Placa PICDEM2+ da Microchip.

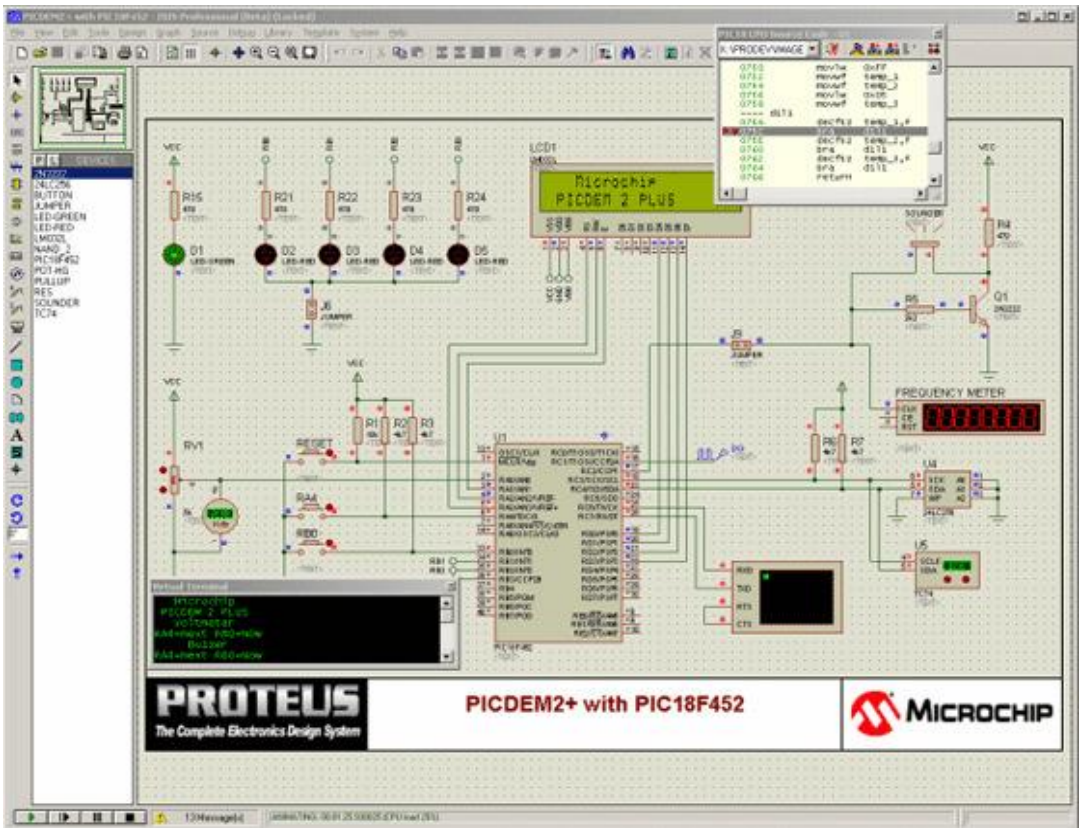


Figura 15.37. Placa PICDEM2+ no ambiente ISIS simulada através do Proteus VSM.

Neste minicurso será usada a versão será usada a versão 7.6 do Proteus.

15.2.3.4. WinPcap

WinPcap é uma biblioteca para monitoramento de tráfego de dados de redes, importante para o funcionamento de uma série de programas e utilitários. A biblioteca ainda oferece funções excelentes para a captura de pacotes, independente da estrutura de rede ou plataforma que esteja em uso. Assim, WinPcap é utilizado como interface de rede de várias ferramentas, como protocolos de análise, monitoramento de redes e sistemas de detecção de rede e tráfego. O usuário necessita executar e instalar este componente para então utilizar tranquilamente qualquer programa que utilize o WinPcap.

Neste minicurso o WinPcap será utilizado como plug-in para o Proteus ISIS, necessário para a simulação da conexão TCP/IP da interface de rede do sistema embarcado construído e simulado no Proteus ISIS. Neste minicurso usamos a versão 4.1.2 desse *software*.

15.2.4. Utilização da Pilha TCP/IP

Nesta seção será discutida a organização e configuração da pilha TCP/IP da Microchip. Caso precisem de suporte e documentação complementar, recomendo buscarem nos seguintes meios:

- Suporte Microchip:
  http://support.microchip.com
- Fórum Microchip:
  http://forum.microchip.com
- Application Note AN833 – The Microchip TCP/IP Stack:
  http://ww1.microchip.com/downloads/en/AppNotes/00833c.pdf
- Microchip TCP/IP Help:
  C:\Microchip Solutions v2010-08-04\Microchip\Help (Diretório da instalação padrão da Pilha)

15.2.4.1. Estrutura de Diretório

No diretório de instalação da Pilha TCP/IP da Microchip será encontrada a estrutura de diretórios apresentada na Figura 15.38.

Os principais subdiretórios são:

Pasta “Microchip”: contém os arquivos fonte da pilha e componentes;

Subpasta “Include”: contém os cabeçalhos (.h) das bibliotecas.

Subpasta “TCPIP Stack”: contém os arquivos C, documentação e utilidades (*software*s).

Subpasta TCPIP Demo App: contém projetos exemplos de utilização da pilha TCP/IP nas placas de desenvolvimento da Microchip.

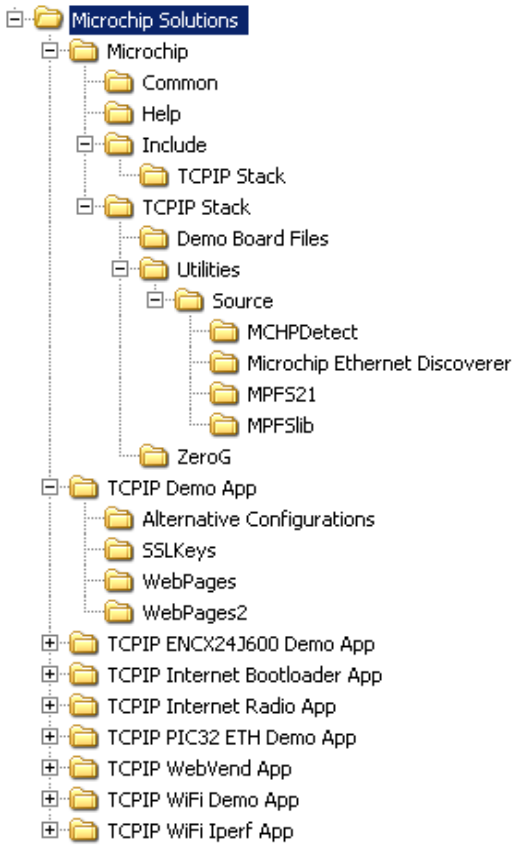


Figura 15.38. Estrutura de diretórios da Pilha TCP/IP da Microchip.

15.2.4.2. *Softwares* Utilitários

Junto com a instalação da Pilha vem uma série de *softwares* utilitários para auxiliar o desenvolvimento de sistemas embarcados. Esses aplicativos podem ser encontrados no diretório: C:\Microchip Solutions v2010-08-04\Microchip\TCPIP Stack\Utilities, ou em Menu Iniciar/Todos os Programas/Microchip/TCPIP Stack v5.25, são eles:

- TCP/IP Configuration Wizard: Permite configurar os parâmetros da Pilha através de um aplicativo gráfico;
- MPFS2 Utility: Utilizado para armazenar as páginas web no sistema embarcado;
- Microchip Web Preview Tool: Permite visualizar as páginas web da forma que serão mostradas pelo sistema embarcado sem que seja necessário carregá-las nele;
- Hash Table Filter Entry Calculator: Esse *software* executa o cálculo CRC sobre seis bytes do endereço de destino em um pacote recebido, em seguida, usa esse valor como um ponteiro para os registradores EHT0-EHT7;
- Ethernet Device Discoverer: Ajuda a descoberta de dispositivos conectados na rede através do protocolo Anounce;
- MCHPDetect: Precursor do Ethernet Device Discoverer.

Neste minicurso usaremos apenas o TCP/IP Configuration Wizard, MPFS2 Utility e Microchip Web Preview Tool. Será apresentado passo-a-passo como usar cada um desses aplicativos.

15.2.4.3. Consumo de Memória

Cada protocolo da Pilha requer certa quantidade de memória de dados e ocupa certa quantidade de memória de programa. Desta forma, de acordo com os protocolos utilizados, podem ser adequados diferentes microcontroladores, com diferentes quantidades de memórias. Na Figura 15.39 é apresentada a quantidade de memória necessária para cada bloco, de forma que o projetista do sistema possa ter uma boa estimativa da quantidade de memória necessária para o desenvolvimento do seu sistema antes de iniciar a implementação.

Module	Program Memory (bytes)- Size Optimization (-0s)	Program Memory (bytes)- Speed Optimization (-03)	Global Data Memory (bytes)
Required Stack Code	9,129	24,069	302
Required Code - ZG2100M Physical Layer	16,074	25,428	458
Announce(1)	+1,872	+2,136	+146
AutoIP	+1,752	+1,917	+48
DHCP Client(1)	+3,894	+4,458	+178
DHCP Server(1)	+3,150	+3,945	+152
DNS Client(1)	+3,345	+4,227	+172
Dynamic DNS Client(1,2,3)	+15,228	+23,676	+434
HTTP2 Server/MPFS2(2)	+20,118	+31,668	+566
ICMP Client	+789	+864	+20
ICMP Server	+345	+372	+0
NBNS(1)	+2,337	+2,652	+146
SMTP Client(1,2,3)	+15,960	+26,520	+486
SNMP Agent(1,4)	+12,486	+21,495	+420
SNTP Client(1,3)	+4,140	+5,208	+198
SSL Client	+23,436	+44,202	+1,354
SSL Server	+24,543	+46,305	+998
SSL Server/Client	+27,210	+53,142	+1,410
TCP	+8,259	+14,979	+176
UDP	+1,374	+1,578	+142
Total(5)	59,598	98,202	1,448
Total + SSL(5)	78,765	N/A(6)	2,684

Notes:

(1): Code size includes the UDP module (required for operation).

(2): Code size includes the TCP module (required for operation).

(3): Code size includes the DNS module (required for operation).

(4): Code size includes the MPFS2 module (required for operation).

(5): Total includes the required files, and the ARP, AutoIP, DHCP Client and Server, DNS, Dynamic DNS, HTTP2, ICMP, MPFS2, NBNS, SMTP, SNMP, SNTP, TCP, and UDP modules.

(6): Compiled code size exceeds the capacity of the target processor.

Figura 15.39. Quantidade de memória de programa necessária para cada protocolo utilizando o compilador MPLAB C30.

15.2.4.4. Placas de Desenvolvimento

A Microchip comercializa uma série de placas de desenvolvimento compatíveis com a pilha TCP/IP. Algumas placas contêm controladores Ethernet e estão prontas para receber o *firmware*, outras precisam de “Placas Filhas” para prover essa funcionalidade.

Nas Figura 15.40, Figura 15.41, Figura 15.42 e Figura 15.43 são mostradas algumas placas de desenvolvimento da Microchip e algumas placas filhas que podem ser conectadas às placas de desenvolvimento para prover interfaces TCP/IP adicionais.

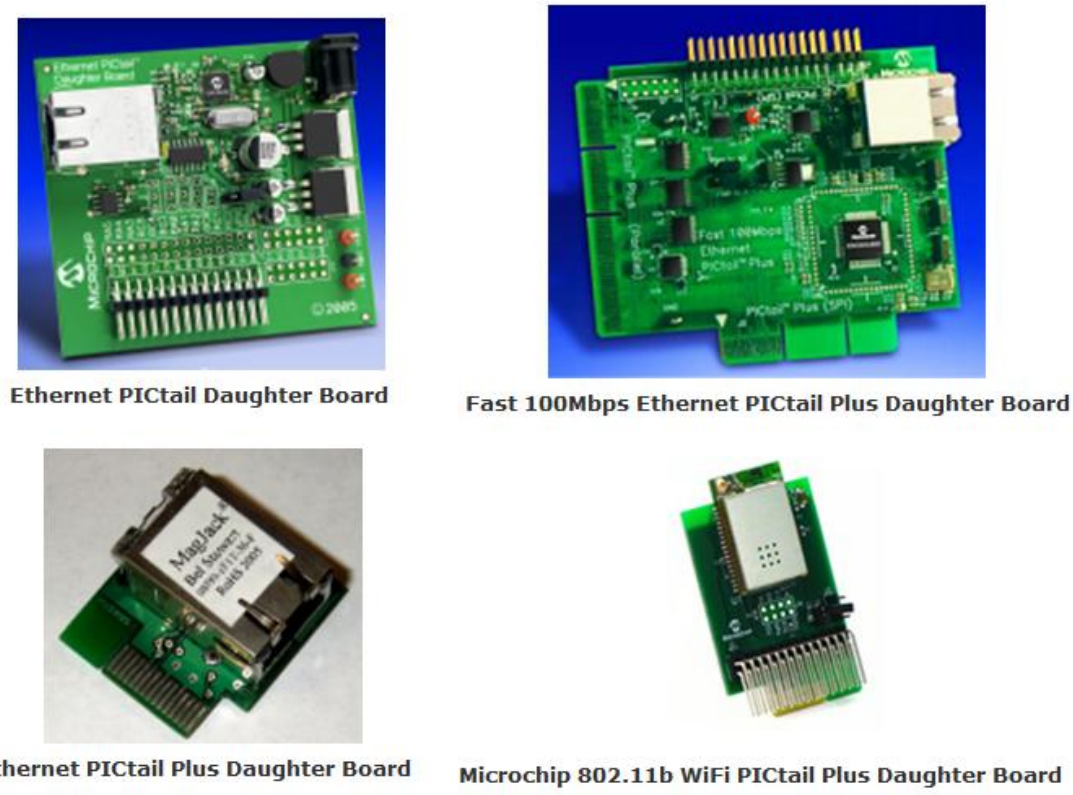


Figura 15.40. Placas "Filhas" Microchip.

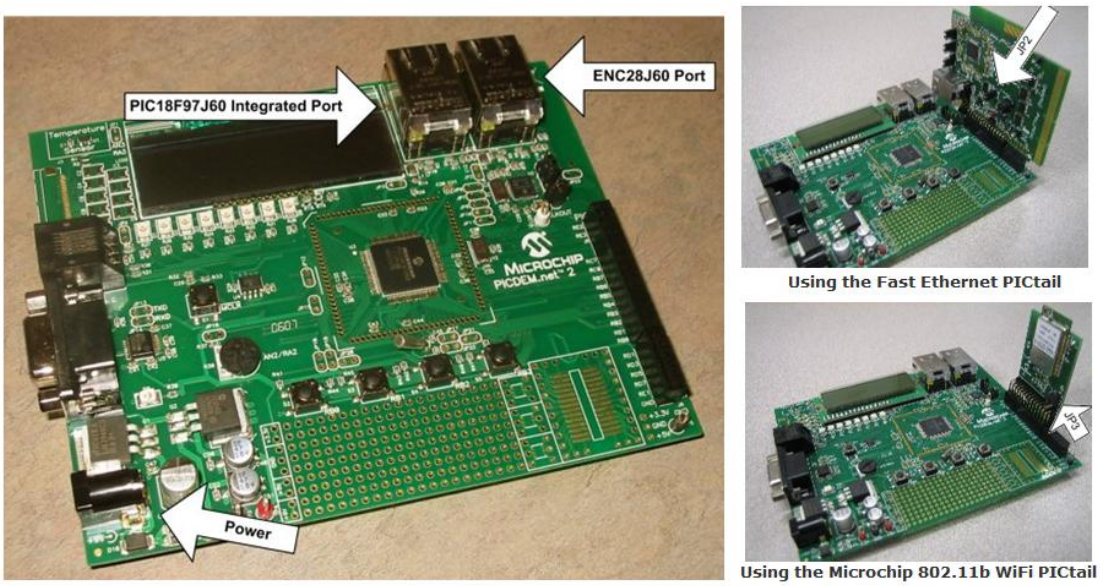


Figura 15.41. Placa de desenvolvimento PICDEM.net 2.

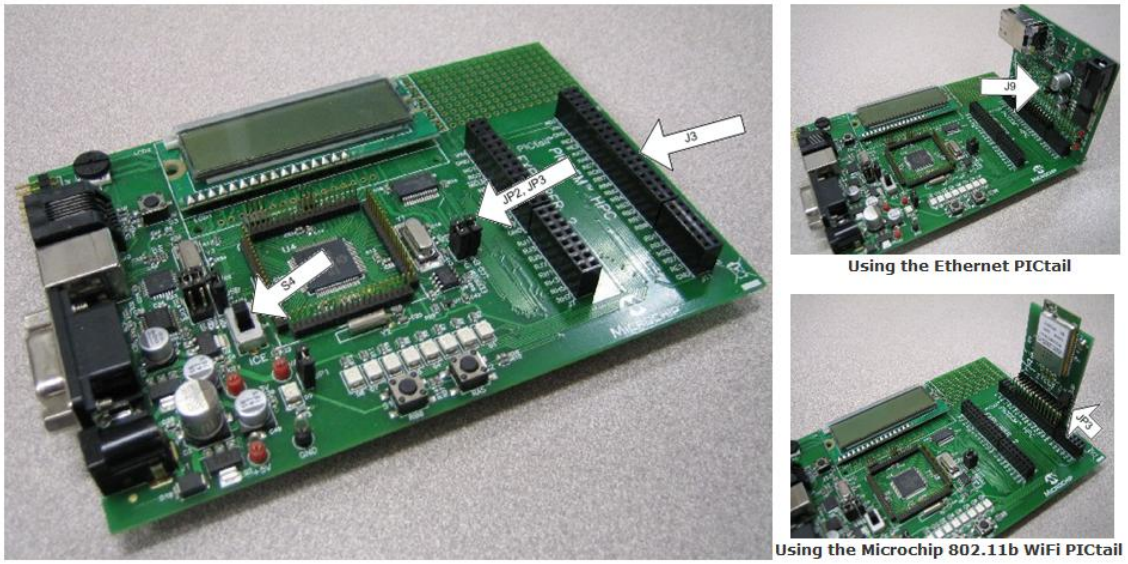


Figura 15.42. Placa de desenvolvimento PIC 18 Explorer.

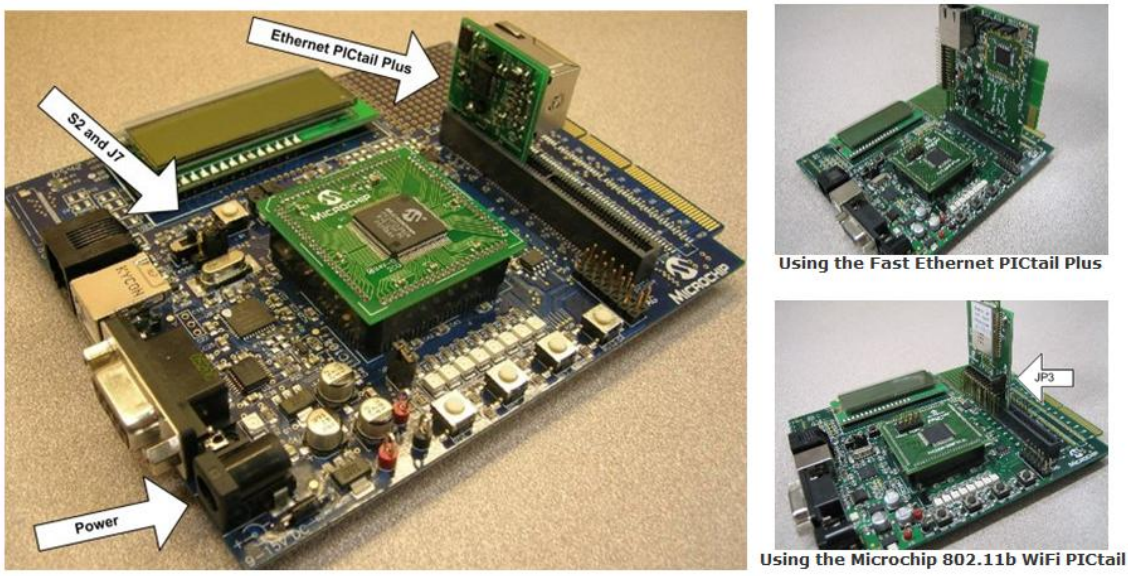


Figura 15.43. Placa de desenvolvimento Explorer 16.

Junto com a pilha TCP/IP vem uma série de projetos exemplos usando as placas de desenvolvimento da Microchip (C:\Microchip Solutions v2010-08-04\TCPIP Demo App). Neste diretório existe o arquivo "TCPIP Demo App-C30 EXPLORER_16 24FJ128GA010 ENC28J60.hex" gerado a partir da compilação do projeto "TCPIP Demo App-C30".

O Proteus ISIS vem com um conjunto de exemplos de placas, entre elas a placa PIC 16 Explorer, apresentada na Figura 15.44. A partir deste projeto, pode-se carregar o arquivo "TCPIP Demo App-C30 EXPLORER_16 24FJ128GA010 ENC28J60.hex" e observar a simulação desta placa.

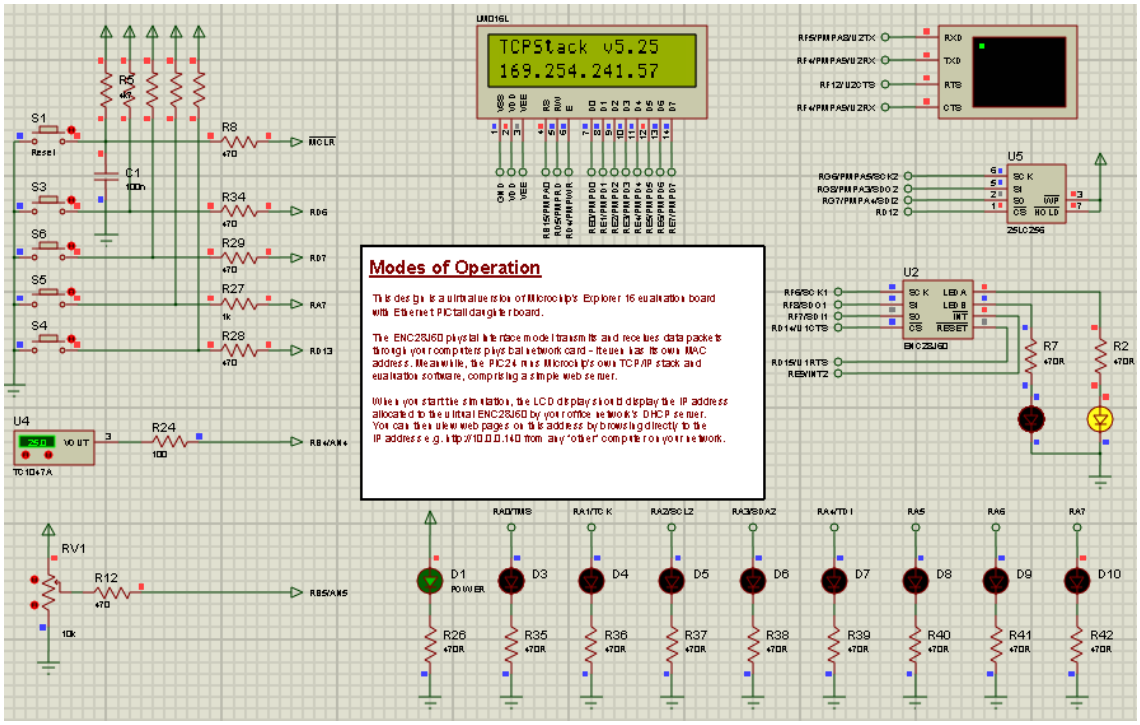


Figura 15.44. Placa Explorer 16 simulada no Proteus - ISIS.

15.2.4.5. Projeto Base

Para facilitar o desenvolvimento de sistemas embarcados usando a Pilha TCP/IP, foi criado um projeto base. Este projeto deve ser editado de forma a compor o sistema desejado. Neste projeto, foi usado o microcontrolador PIC 24F128GA006, o controlador ethernet ENC28J60 e uma memória EEPRON 25LC256. O circuito do Projeto Base simulado no Proteus-ISIS está apresentado na Figura 15.45.

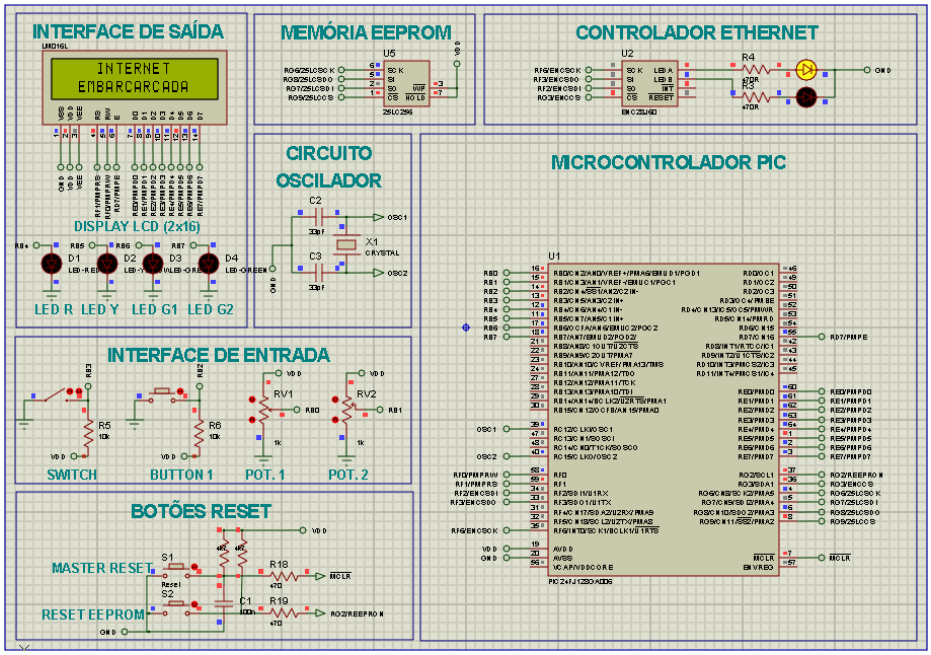


Figura 15.45. Circuito do Projeto Base.

O sistema desenvolvido no Projeto Base é capaz de monitorar uma chave, um botão e dois potenciômetros, exibindo seus valores na Web. O sistema também é equipado com quatro LEDs, um vermelho, um amarelo e dois verdes. O LED vermelho é aceso quando a chave é fechada e o botão pressionado, o LED amarelo é aceso quando o valor do potenciômetro 1 ultrapassa 50%. A placa também é equipada com um display LCD de duas linhas e dezesseis colunas que exibem mensagens enviadas pela Web, o valor do botão, da chave e dos potenciômetros.

O código fonte do *firmware* do projeto base pode ser encontrado no seguinte diretório do CD do minicurso: “Projeto Base\Firmware\Projeto Base\TCPIP Demo App-C30”.

Neste projeto estão contidos os seguintes arquivos da Pilha TCP/IP Microchip:

- TCPIP.h: Controla todo o funcionamento dos protocolos pilha TCP/IP;
- IP.c e IP.h: Implementação do protocolo IP;
- TCP.c e TCP.h: Implementação do protocolo TCP;
- MAC.h: Implementação do protocolo MAC;
- ARP.c e ARP.h: Implementação do protocolo ARP;
- Tick.c e Tick.h: Define a referência de temporização da pilha;
- Delay.c e Delay.h: Implementação dos temporizadores usados pela pilha;
- Helpers.ceHelpers.h: Várias funções auxiliares usadas pela pilha;
- StackTsk.c e StackTsk.h: Coordena a chamada das tarefas da pilha;
- MainDemo.c e MainDemo.h: Implementação das funcionalidades do sistema embarcado (leitura de sensores e botões, exibição de informações em display etc);
- *Hardware*Profile.h: Configuração do *Hardware* do sistema. Neste arquivo, as seguintes atividades são realizadas: a configuração dos módulos do microcontrolador (*watchdog*, osciladores, JTAG, etc), a definição do *clock* do sistema, a definição dos pinos de conexão do controlador Ethernet, da memória EEPROM e do display LCD e a configuração dos pinos de entrada e saídas analógicas e digitais;
- TCPIPConfig.h: Configuração da pilha TCP/IP. Neste arquivo, são definidos: os protocolos usados (ICMP, HTTP, SMTP, etc.); onde será feito o armazenamento de dados (memória EEPROM ou memória Flash); o sistema de arquivos usado (MPFS ou MPFS2); IP, Máscara, Gateway, DNS e *host*name; endereço MAC; opções da camada de transporte (TCP e UDP); configurações específicas das aplicações;
- MPFS2.c e MPFS2.h: Implementação do sistema de arquivos da Microchip para microcontroladores PIC;
- HTTP2.c e HTTP2.h: Implementação do servidor HTTP;
- ICMP.c e ICMP.h: Implementação do protocolo ICMP, usado para responder ao comando PING;

- DHCP.c e DHCP.h: Implementação do protocolo DHCP, usado para a definição automática do IP, máscara de rede, gateway e servidores DNS de cada estação.
- HTTPPrint.h: Definição das variáveis dinâmicas das páginas web;
- CustomHTTPApp.c: Impelmentação das funções de retorno das variáveis dinâmicas das páginas web exibidas pelo servidor HTTP2 e implementação da função de retorno GET.

No desenvolvimento de sistemas embarcados simples os únicos arquivos dentre os listados que precisam ser alterados são: HardwareProfile.h, TCPIPConfig.h e main.c.

15.2.4.5.1. HardwareProfile.h

Neste arquivo é feita toda a configuração de *hardware* do sistema. Na Figura 15.46 é apresentado a primeira parte desse arquivo. Neste trecho são definidas as configurações de *hardware* como: tipo de oscilador, ativação de periféricos e funcionalidades como o *watchdog*, entre inúmeras outras opções. Nas linhas 30, 31 e 32 é definida a frequência de *clock*, usada como base de temporização para toda a Pilha TCP/IP.

```
16      #ifndef __HARDWARE_PROFILE_H
17      #define __HARDWARE_PROFILE_H
18
19      #include "GenericTypeDefs.h"
20      #include "Compiler.h"
21
22      // Set configuration fuses
23      #if defined(THIS_IS_STACK_APPLICATION)
24          _CONFIG2(FNOSC_PRIPLL & POSCMOD_XT)     // Primary XT OSC with 4x PLL
25          _CONFIG1(JTAGEN_OFF & FWDTEN_OFF)       // JTAG off, watchdog timer off
26      #endif // Prevent more than one set of config fuse definitions
27
28          // Clock frequency value.
29          // This value is used to calculate Tick Counter value
30          #define GetSystemClock()        (32000000ul)      // Hz
31          #define GetInstructionClock()   (GetSystemClock()/2)
32          #define GetPeripheralClock()    GetInstructionClock()
```

Figura 15.46. HardwareProfile.h, configuração de *hardware* e definidas a frequência do sistema.

No arquivo HardwareProfile.h também são definidos pinos do microcontrolador que serão conectados ao controlador Ethernet ENC28JC, à memória EEPROM 25LC256, e ao display LCD 2x16. Os códigos que descrevem essas ligações são exibidos nas Figura 15.47, Figura 15.48 e Figura 15.49.

```
65      // ENC28J60 I/O pins
66      #define ENC_CS_TRIS         (TRISGbits.TRISG3)  // Comment this
67      #define ENC_CS_IO           (LATGbits.LATG3)
68      // SPI SCK, SDI, SDO pins are automatically controlled by the
69      // PIC24/dsPIC/PIC32 SPI module
70      #define ENC_SPI_IF          (IFS0bits.SPI1IF)
71      #define ENC_SSPBUF          (SPI1BUF)
72      #define ENC_SPISTAT         (SPI1STAT)
73      #define ENC_SPISTATbits     (SPI1STATbits)
74      #define ENC_SPICON1         (SPI1CON1)
75      #define ENC_SPICON1bits     (SPI1CON1bits)
76      #define ENC_SPICON2         (SPI1CON2)
```

Figura 15.47. HardwareProfile.h, configuração dos pinos do controlador ethernet ENC28J60.

```
78      // 25LC256 I/O pins
79      #define EEPROM_CS_TRIS      (TRISGbits.TRISG9)
80      #define EEPROM_CS_IO        (LATGbits.LATG9)
81      #define EEPROM_SCK_TRIS     (TRISGbits.TRISG6)
82      #define EEPROM_SDI_TRIS     (TRISGbits.TRISG7)
83      #define EEPROM_SDO_TRIS     (TRISGbits.TRISG8)
84      #define EEPROM_SPI_IF       (IFS2bits.SPI2IF)
85      #define EEPROM_SSPBUF       (SPI2BUF)
86      #define EEPROM_SPICON1      (SPI2CON1)
87      #define EEPROM_SPICON1bits  (SPI2CON1bits)
88      #define EEPROM_SPICON2      (SPI2CON2)
89      #define EEPROM_SPISTAT      (SPI2STAT)
90      #define EEPROM_SPISTATbits  (SPI2STATbits)
```

Figura 15.48. HardwareProfile.h, configuração dos pinos da memória EEPROM 25LC256.

```
95      // LCD Module I/O pins.
96      #define LCD_DATA_TRIS       (*((volatile BYTE*)&TRISE))
97      #define LCD_DATA_IO         (*((volatile BYTE*)&LATE))
98      #define LCD_RD_WR_TRIS      (TRISFbits.TRISF0)
99      #define LCD_RD_WR_IO        (LATFbits.LATF0)
100     #define LCD_RS_TRIS         (TRISFbits.TRISF1)
101     #define LCD_RS_IO           (LATFbits.LATF1)
102     #define LCD_E_TRIS          (TRISDbits.TRISD7)
103     #define LCD_E_IO            (LATDbits.LATD7)
```

Figura 15.49. HardwareProfile.h, configuração dos pinos do módulo LCD.

No arquivo HardwareProfile.h também são definidos os pinos de estrada e saída dos periféricos. Nas Figura 15.50 é exibido o código que define os pinos de entrada para dois botões, uma chave e dois potenciômetros. Na Figura 15.51 é exibido o código que define os pinos de saída para 4 LEDs.

Para fazer uma modificação, o único trecho de código que precisa ser modificado é o que contém os "*defines*" que definem os pinos de entrada de saída, desde que a conexão do controlador ethernet, memória EEPROM e display LCD mantenham-se inalterados.

```
34          // Button EEPRON Reset
35          #define BUTTON0_TRIS        (TRISGbits.TRISG2)
36          #define BUTTON0_IO          (PORTGbits.RG2)
37
38          // Switch Button
39          #define SWITCH_TRIS         (TRISBbits.TRISB3)
40          #define SWITCH_IO           (PORTBbits.RB3)
41
42          // Button 1
43          #define BUTTON1_TRIS        (TRISBbits.TRISB2)
44          #define BUTTON1_IO          (PORTBbits.RB2)
45
46          // Potenciometer 1
47          #define POT1_TRIS           (TRISBbits.TRISB0)
48          #define POT1_IO             (PORTBbits.RB0)
49          // Potenciometer 2
50          #define POT2_TRIS           (TRISBbits.TRISB1)
51          #define POT2_IO             (PORTBbits.RB1)
52
```

Figura 15.50. HardwareProfile.h, definição dos periféricos de entrada.

```
53          // Led Red
54          #define LED_R_TRIS          (TRISBbits.TRISB4)
55          #define LED_R_IO            (PORTBbits.RB4)
56          // Led Yellow
57          #define LED_Y_TRIS          (TRISBbits.TRISB5)
58          #define LED_Y_IO            (PORTBbits.RB5)
59          // Leds Green
60          #define LED_G1_TRIS         (TRISBbits.TRISB6)
61          #define LED_G1_IO           (PORTBbits.RB6)
62          #define LED_G2_TRIS         (TRISBbits.TRISB7)
63          #define LED_G2_IO           (PORTBbits.RB7)
```

Figura 15.51. *Hardware*Profile.h, definição dos periféricos de saída.

15.2.4.5.2. TCPIPConfig.h

Neste arquivo é feita a configuração da pilha TCP/IP da microchip, na qual são definidos:

- Os protocolos usados (ICMP, HTTP, SMTP, etc.);
- Onde será feito o armazenamento de dados (memória EEPROM ou memória Flash);
- Sistema de arquivos usado (MPFS ou MPFS2);
- IP, Máscara, Gateway, DNS, *hostname*;
- MAC;
- Opções da camada de transporte (TCP e UDP);
- Configurações específicas das aplicações.

Nas Figura 15.52, Figura 15.53, Figura 54 é apresentado o código do arquivo TCPIPConfig.h. Para facilitar a configuração deste arquivo pode ser usado o aplicativo TCP/IP Configuration Wizard. Ele fornece uma interface gráfica simples que permite a seleção e configuração dos protocolos desejados. Além disso, ele gera o arquivo TCPIPConfig.h.

```
/* Application Level Module Selection
 *   Uncomment or comment the following lines to enable or
 *   disabled the following high-level application modules.
 */
//#define STACK_USE_UART                  // Application demo using UART for IP address display and stack
//#define STACK_USE_UART2TCP_BRIDGE     // UART to TCP Bridge application example
//#define STACK_USE_IP_GLEANING
#define STACK_USE_ICMP_SERVER           // Ping query and response capability
//#define STACK_USE_ICMP_CLIENT         // Ping transmission capability
//#define STACK_USE_HTTP_SERVER         // Old HTTP server
#define STACK_USE_HTTP2_SERVER          // New HTTP server with POST, Cookies, Authentication, etc.
//#define STACK_USE_SSL_SERVER          // SSL server socket support (Requires SW300052)
//#define STACK_USE_SSL_CLIENT          // SSL client socket support (Requires SW300052)
//#define STACK_USE_AUTO_IP               // Dynamic link-layer IP address automatic configuration protocol
#define STACK_USE_DHCP_CLIENT           // Dynamic Host Configuration Protocol client for obtaining IP addre
```

Figura 15.52. TCPIPConfig.h, habilitação dos Módulos (HTTP, ICMP, DHCP etc).

```
#define MY_DEFAULT_HOST_NAME            "BACURAU"

#define MY_DEFAULT_MAC_BYTE1            (0x00)  // Use the default of
#define MY_DEFAULT_MAC_BYTE2            (0x00)  // 00-04-A3-00-00-00 if using
#define MY_DEFAULT_MAC_BYTE3            (0x00)  // an ENCX24J600 or MRF24WB0M
#define MY_DEFAULT_MAC_BYTE4            (0x00)  // and wish to use the internal
#define MY_DEFAULT_MAC_BYTE5            (0x00)  // factory programmed MAC
#define MY_DEFAULT_MAC_BYTE6            (0x01)  // address instead.
```

Figura 15.53. TCPIPConfig.h, definição de *host*name e MAC.

```
#define MY_DEFAULT_IP_ADDR_BYTE1        (169ul)
#define MY_DEFAULT_IP_ADDR_BYTE2        (254ul)
#define MY_DEFAULT_IP_ADDR_BYTE3        (1ul)
#define MY_DEFAULT_IP_ADDR_BYTE4        (1ul)

#define MY_DEFAULT_MASK_BYTE1           (255ul)
#define MY_DEFAULT_MASK_BYTE2           (255ul)
#define MY_DEFAULT_MASK_BYTE3           (0ul)
#define MY_DEFAULT_MASK_BYTE4           (0ul)

#define MY_DEFAULT_GATE_BYTE1           (169ul)
#define MY_DEFAULT_GATE_BYTE2           (254ul)
#define MY_DEFAULT_GATE_BYTE3           (1ul)
#define MY_DEFAULT_GATE_BYTE4           (1ul)

#define MY_DEFAULT_PRIMARY_DNS_BYTE1    (169ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE2    (254ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE3    (1ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE4    (1ul)

#define MY_DEFAULT_SECONDARY_DNS_BYTE1  (0ul)
#define MY_DEFAULT_SECONDARY_DNS_BYTE2  (0ul)
#define MY_DEFAULT_SECONDARY_DNS_BYTE3  (0ul)
#define MY_DEFAULT_SECONDARY_DNS_BYTE4  (0ul)
```

Figura 54. TCPIPConfig.h, definição do IP, Máscara, Gatway e Servidor DNS.

15.2.4.5.3. Main.c

O arquivo main.c é constituído basicamente de duas partes, Inicialização e Loop. Na inicialização é feita todas as configurações do *hardware* e da Pilha, seguindo a seguinte sequência:

1. Configuração do *Hardware* (pinos de E/S, oscilador, A/D, módulo SPI etc);
2. Inicialização do temporizador (TickInit(););
3. Inicialização do sistema de arquivos (MPFSInit(););
4. Inicialização da estrutura APP_CONFIG InitAppConfig();
5. Inicialização dos módulos da pilha TCP/IP (StackInit();).

Após a inicialização, o sistema entra em um loop infinito onde é feito o envio e recebimento de dados via rede, leitura de dados dos botões, chaves e potenciômetros, bem como o envio de dados para o display e LEDs. No Loop são feitas chamadas as seguintes funções na seguinte sequência:

1. StackTask(): executa as operações temporizadas e coordena o recebimento e o envio de pacotes;
2. StackApplications(): executa os módulos carregados (Servidor HTTP2);
3. ProcessIO(): processa os dados de entrada e saída através da técnica de Cooperativismo Multitarefa

15.2.4.5.3.1 Cooperativismo Multitarefa

Em um sistema cooperativo multitarefa as tarefas são executadas sequencialmente e de modo independente. O programa principal envia cada tarefa para ser processada e, ao final da sua execução, o controle retorna ao programa principal e só então a próxima tarefa fica apta a ser executada (MICROCHIP, 2008b).

Geralmente, o cooperativismo multitarefa é implementado pelo sistema operacional ou pela própria aplicação principal. A pilha TCP/IP da Microchip foi desenvolvida para ser independente de qualquer sistema operacional e, por isso, ela implementa seu próprio mecanismo cooperativo multitarefa. Dessa forma, ela pode ser utilizada em qualquer sistema, independente se o mesmo utiliza o sistema cooperativo multitarefa ou não. De qualquer forma, uma aplicação que utiliza a pilha TCP/IP da Microchip deve também utilizar o método cooperativo multitarefa. Isso é feito através da divisão de um programa em múltiplas tarefas, ou organizando o programa principal como uma máquina de estados finita (FSM) e dividindo um programa grande em diversos programas pequenos (MICROCHIP, 2008b).

Na prática o cooperativismo é implementado da seguinte forma: na função *main* as atividades a serem desenvolvidas são divididas em diversas tarefas e cada tarefa colocada em um “case” de um “switch”. A variável de controle do “switch” é uma variável do tipo global e armazena o próximo estado. Após finalizar a execução de uma tarefa, essa variável é atualizada e a próxima tarefa será executada na próxima execução do loop. Na figura é ilustrada a estrutura da função main.c.

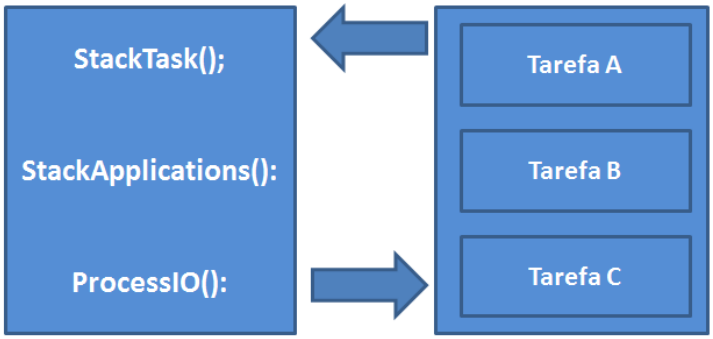


Figura 15.55. Estrutura da main.c construída sobre cooperativismo multitarefa.

15.2.4.6. Monitoramento Web

Nesta seção serão apresentados os passos necessários para a construção de um servidor web embarcado para monitoramento de sensores. Este tipo de sistema pode ser usado, por exemplo, para monitorar, via web, a temperatura de uma sala, a vibração de máquinas em uma indústria, o nível de um reservatório de água, etc.

Neste tipo de sistema haverá uma ou mais variáveis dinâmicas nas páginas web do sistema embarcado. Essas variáveis serão atualizadas dinamicamente pelo servidor web embarcado, permitindo o monitoramento dessas grandezas em tempo real. As variáveis dinâmicas são indicadas no código HTML das páginas web armazenadas no servidor entre tiles. Quando uma página que contiver uma variável dinâmica for acessada, o sistema faz uma chamada a uma função de retorno que substitui o nome da variável entre tiles pelo seu valor atual.

Por exemplo, para criar uma variável dinâmica chamada VAR deve-se:

1. Inserir ~VAR~ na página web (HTML) no local onde deseja que esse valor seja exibido;
2. Implementar a função de retorno HTTPPrint_var() em CustomHTTPApp.c, conforme será visto a seguir.

A página web de monitoramento do Projeto Base, cujo circuito foi mostrado na Figura 15.45, é mostrada na Figura 15.56. Na parte superior da página é exibido o *Hostname* e o endereço IP da placa acessada. Em seguida, são exibidas as informações dinâmicas monitoradas (estados dos LEDs, da chave, botão e potenciômetros). O código HTML desta página de monitoramento é exibido na Figura 15.57.

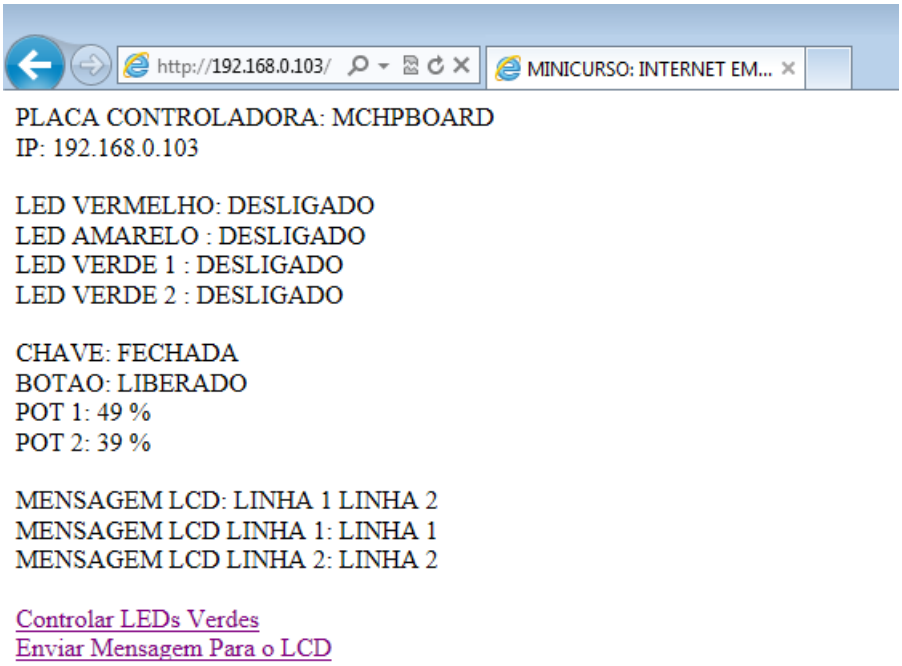


Figura 15.56. Página web de monitoramento com variáveis dinâmicas.

```
PLACA CONTROLADORA: ~HOSTNAME~
<br />IP: ~IP~
<br />
<br />LED VERMELHO: ~LED(1)~
<br />LED AMARELO : ~LED(2)~
<br />LED VERDE 1 : ~LED(3)~
<br />LED VERDE 2 : ~LED(4)~
<br />
<br />CHAVE: ~SWITCH~
<br />BOTAO: ~BUTTON~
<br />POT 1: ~POT(1)~ %
<br />POT 2: ~POT(2)~ %
<br />
<br />MENSAGEM LCD: ~LCD~
<br />MENSAGEM LCD LINHA 1: ~LCD1~
<br />MENSAGEM LCD LINHA 2: ~LCD2~
<br />
```

Figura 15.57. Código HTML da página de monitoramento.

Observa-se que cada variável dinâmica possui um nome entre tiles. Para cada uma dessas variáveis dinâmicas deve ser criada uma função de retorno no arquivo CustomHTTPApp.c com o nome HTTPPrint_NOME(), onde NOME deve ser substituído pelo nome da variável dinâmica. Sendo assim, as seguintes funções devem ser criadas e implementadas:

- void HTTPPrint_IP(void);
- void HTTPPrint_*HOST*NAME(void);
- void HTTPPrint_LED(WORD);
- void HTTPPrint_SWITCH(void);

- void HTTPPrint_BUTTON(void);
- void HTTPPrint_POT(WORD);
- void HTTPPrint_LCD(void);
- void HTTPPrint_LCD1(void);
- void HTTPPrint_LCD2(void);

Segue abaixo o código da função HTTPPrint_*HOST*NAME(void):

```
void HTTPPrint_HOSTNAME(void)
{
	TCPPutString(sktHTTP, AppConfig.NetBIOSName);
}
```

Essa função não recebe nenhum parâmetro nem retorna valor. Simplesmente é feita uma chamada a função TCPPutString, que recebe como parâmetro o ponteiro do socket TCP para escrita na página HTML (sktHTTP) e o nome da variável que se deseja retornar. A variável NetBIOSName da estrutura AppConfig é do tipo BYTE = unsigned char (definido no arquivo GenericTypeDefs.h).

Quando se deseja substituir a variável dinâmica por uma *string* que não esteja armazenada em uma variável, deve ser usada a função TCPPutROMString em vez de TCPPutString. A *string* a ser enviada deve ser precedida do *cast* (ROM BYTE*). Na Função HTTPPrint_BUTTON(void) podemos ver um exemplo de utilização desta função:

```
void HTTPPrint_BUTTON(void){
	if(!BUTTON1_IO)
		TCPPutROMString(sktHTTP, (ROM BYTE*)"PRESSIONADO");
	else
		TCPPutROMString(sktHTTP, (ROM BYTE*)"LIBERADO")
}
```

Esta função retorna a *string* “PRESSIONADO” caso o botão esteja pressionado e “LIBERADO” se o botão não estiver pressionado.

Caso a *string* retornada para a variável tenha mais de 16 caracteres outro procedimento deve ser feito. Usaremos a função HTTPPrint_LCD, apresentada abaixo, para apresentar esse procedimento.

```
void HTTPPrint_LCD(void)
{
	if(strlen((char*)displayLine12)>TCPIsPutReady(sktHTTP))
	{// Not enough space
		curHTTP.callbackPos = 0x01;
		return;
	}
	TCPPutString(sktHTTP, (BYTE*)displayLine12);
	curHTTP.callbackPos = 0x00;
```

```
        return;
}
```

Inicialmente, é testado se o tamanho da *string* a ser enviada, neste caso a *string* displayLine12, é maior do que o soquete pode enviar (16 caracteres). Se o tamanho for maior, a variável curHTTP.callbackPos é atualizada com o valor 0x01, indicando que mais 2 bytes (16 caracteres) devem ser alocados para o envio da *string*. Em seguida, a função será chamada novamente, e caso haja espaço suficiente a *string* é retornada e a variável curHTTP.callbackPos atualizada com o valor 0x00.

Observa-se na Figura 15.57 que as variáveis dinâmicas referentes aos valores dos potenciômetros e estados dos LEDs, respectivamente POT e LED, são definidas com o mesmo nome, distinguidas entre si por índices. Neste caso, somente uma função de retorno deve ser implementada no arquivo CustomHTTPApp.c, porém, ao contrário das variáveis não indexadas, essa função deve receber um parâmetro do tipo WORD (*unsigned short int*, 16 bits sem sinal). A partir do valor desse parâmetro, a variável dinâmica pode ser univocamente identificada e seu respectivo valor retornado. O código da função HTTPPrint_POT, que faz uso desse artifício, é apresentado abaixo.

```
void HTTPPrint_POT(WORD param)
{
        BYTE String[16];
        switch(param)
        {
                        case 1:  uitoa(potenciometer1, (BYTE*)String);
                                 TCPPutString(sktHTTP, String);
                                 break;
                case 2:  uitoa(potenciometer2, (BYTE*)String);
                                 TCPPutString(sktHTTP, String);
                                 break;
        }
        return;
}
```

Para que o sistema funcione, além de definir as funções de retorno no arquivo CustomHTTPApp.c, é preciso definir seus protótipos em HTTPPrint.h. Este arquivo pode ser gerado automaticamente pelo *software* utilitário MPFS Generator.

O MPFS Generator é uma aplicação gráfica utilizada para construir uma imagem MPFS ou MPFS2 a ser armazenada no microcontrolador ou em uma memória externa serial EEPROM. Esta imagem é obtida a partir de um conjunto de arquivos que, em geral, são páginas HTML, imagens, dentre outros. O MPFS2 Utility transforma as páginas web em um formato eficiente para armazenamento em um sistema embarcado. Desta forma, além de gerar o arquivo de cabeçalho contendo os protótipos das funções de retorno, este *software* gera o sistema de arquivos contendo as páginas web armazenado na memória

EEPROM. A tela do MPFS Generator com as configurações que devem ser executadas está apresentada na Figura 15.58.

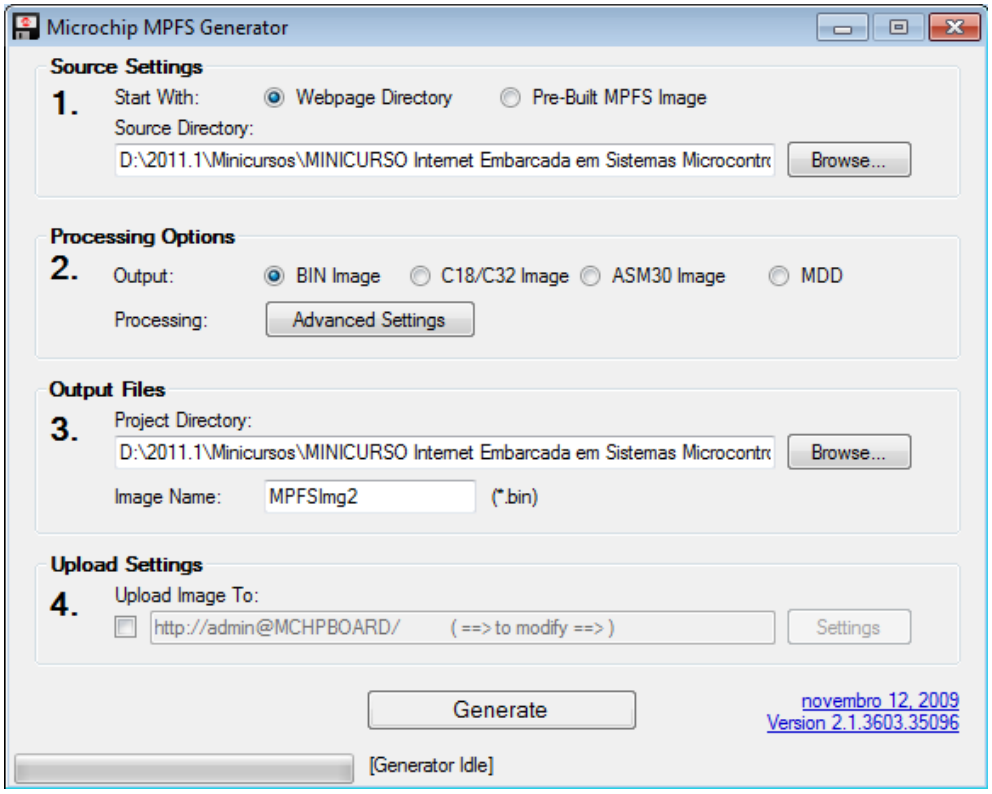


Figura 15.58. Tela do MPFS Generator.

No passo 1 deve ser selecionado o diretório onde se encontram as páginas web. No passo 2 é definido o tipo de imagem, a imagem binária deve ser selecionada. No passo 3 é definido o diretório e o nome do arquivo contendo a imagem a ser gerada. O passo 4 permite o upload das páginas geradas para o sistema embarcado, porém não utilizaremos essa funcionalidade. Ao executar todas as configurações e clicar em *Generate*, a imagem contendo as páginas web será gravada no diretório de saída.

Nós já vimos como criar as páginas web dinâmicas, implementar as funções de retorno no sistema embarcado e gerar o arquivo com a imagem das páginas web. Porém como gravar esse arquivo na memória EEPROM do sistema? Via rede, acessando o endereço: http://endereco_ip/mpfsupload, no qual o endereço_ip deve ser substituído pelo IP da placa do servidor web. A tela apresentada na Figura 15.59 será exibida. O usuário deve clicar no botão Procurar, selecionar o arquivo gerado anteriormente pelo MPFS Generator e em seguida clicar no botão Upload. Após o envio do arquivo, a seguinte mensagem será exibida “MPFS Image Successful”.

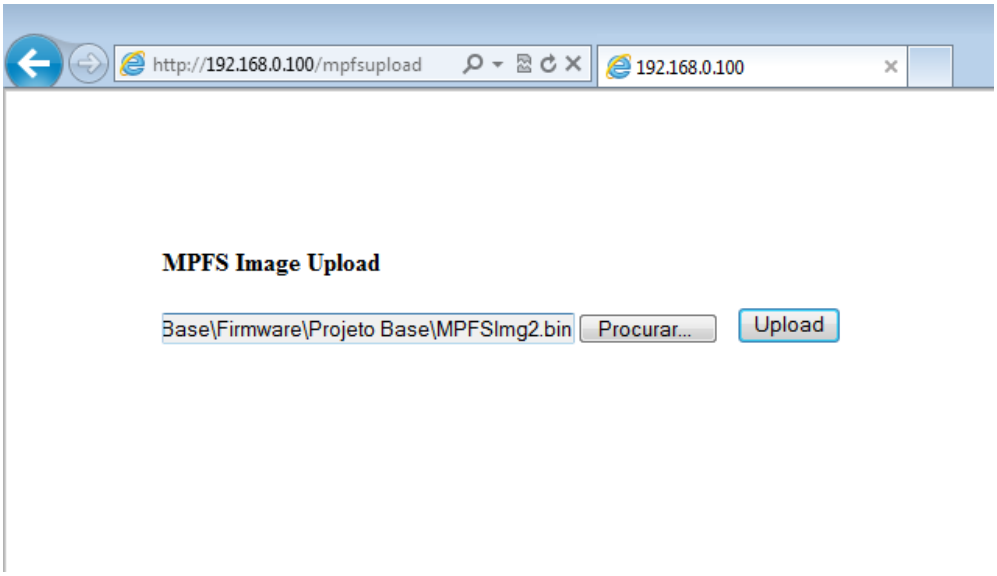


Figura 15.59. Página Web do MPFS Image Upload.

15.2.4.7. Controle Web

Na seção anterior foi apresentado como construir um sistema embarcado para monitoramento via web, ou seja, para visualização de dados do sistema embarcado através de um navegador web. Nesta seção serão apresentados os passos necessários para a construção de um servidor web para controle do sistema embarcado.

Para o envio de dados via páginas HTML são usados formulários. Os formulários HTML são estruturas que permitem o envio de dados de um cliente web (*browser*) para um servidor web. Formulários HTML são definidos pela tag <*form*> e contém um ou mais campos *input* (*text fields*, *checkboxes*, *radio-buttons*, *submit button*, entre outros).

Nos formulários HTML também é definido o método de envio de dados, que pode ser GET ou POST. No método GET o dado é enviado junto com a URL.Neste caso, os dados são mais fáceis de ser processados do que no método POST, porém limitados ao tamanho máximo de 100 bytes. No método POST os dados são enviados no corpo da mensagem e não há limitação de tamanho.

Apesar de o servidor HTTP da Microchip suportar o uso do método POST, neste documento será apresentado somente o uso do método GET, pelo fato de ser mais simples. Abaixo é apresentado um trecho do código HTML para o envio de duas mensagens de texto será serem exibidas no sistema embarcado, uma em cada linha do display LCD de duas linhas por dezesseis colunas. A página web gerada a partir desse código é mostrada na Figura 15.60.

```
<form method="get" action="formLCD.htm">
    <div id="formulario3">
      <br />Mensagem Linha 1 LCD:<input name="MSG1" type="text">
      <br />Mensagem Linha 2 LCD:<input name="MSG2" type="text">
      <br />(No maximo 16 caracteres por linha)
```

```
        <br /><br /><input type="submit" value="Enviar Mensagem">
    </div>
</form>
```

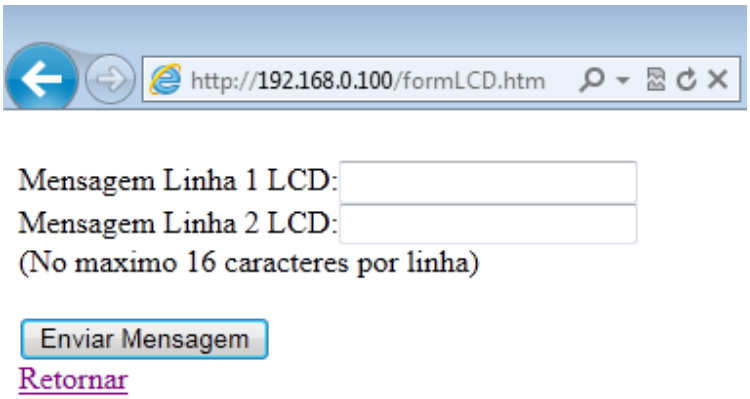


Figura 15.60. Formulário para envio das mensagens para os displays LCD.

Na primeira linha do código HTML é definido o método de envio do formulário. Neste caso, GET é a ação a ser executada quando o formulário for enviado. No campo *action* deve ser colocado entre aspas duplas o nome do arquivo HTML que contém o formulário. Ao clicar no botão "Enviar Mensagem" as mensagens nos campos de entrada MSG1 e MSG2 são enviadas para o servidor que hospeda a página web. Os dados dos campos de entrada são enviados em pares nome/valor, separados por "&". Se no exemplo anterior, o usuário digitar na primeira caixa de texto a *string* "mensagem_1" e na segunda a *string* "mensagem_2", ao clicar em enviar mensagem, os dados serão enviados pela URL da seguinte forma: "/formLCD.htm?MSG1=mensagem_1&MSG2=mensagem_2".

Após a codificação dos formulários nos arquivos HTML, deve ser implementada a função HTTPExecuteGet() no arquivo CustomHTTPApp.c. Essa função é responsável pelo recebimento dos dados via comando GET. Seu código é apresentado na Figura 15.61. O nome do formulário que executou o envio é obtido pela função MPFSGetFilename. Por exemplo, no trecho de código: "MPFSGetFilename (curHTTP.file, filename, 20);" o nome do formulário que executou o envio é armazenado na variável *filename*. O valor de uma variável é obtido pela função HTTPGetROMArg. Por exemplo, no trecho de código "ptr=HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"MSG1");" o valor da variável MSG1 é armazenado na variável *ptr*.

```
154     HTTP_IO_RESULT HTTPExecuteGet(void)
155     {
156         BYTE *ptr;
157         BYTE filename[20];
158
159         // Load the file name
160         // Make sure BYTE filename[] above is large enough for your longest name
161         MPFSGetFilename(curHTTP.file, filename, 20);
162
163         else
164         // If its the formLCD.htm page
165         if(!strcmppgm2ram((char*)filename, (ROM char*)"formLCD.htm"))
166         {
167             // Find the new MSG1 value
168             ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"MSG1");
169             if(ptr){
170                 strcpypgm2ram(displayLine1,(char*)ptr);
171             }
172
173             // Find the new MSG2 value
174             ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"MSG2");
175             if(ptr){
176                 strcpypgm2ram(displayLine2,(char*)ptr);
177             }
178
179             // Updates the displayLine12
180             strcpypgm2ram(displayLine12, displayLine1);
181             strcat(displayLine12, displayLine2);
182         }
183
184         return HTTP_IO_DONE;
185     }
```

Figura 15.61. Códificação da função HTTPExecuteGet.

15.3 Referências

BALTAZAR, J. A. **Acionamento de cargas e leitura de sensores utilizando redes Ethernet.** Viçosa, MG. Universidade Federal de Viçosa. 2008.

MICROCHIP. **ENC28J60 Data Sheet.** 2008a. Disponível em <http://ww1.microchip.com/downloads/en/DeviceDoc/39662c.pdf>. Acessado em 10 Nov. 2010.

MICROCHIP. **Microchip TCP/IP Stack Application Note**. 2008b. Disponível em <http://ww1.microchip.com/downloads/en/AppNotes/00833c.pdf>. Acessado em 10 Nov. 2010.

MICROCHIP. **MPLAB Integrated Development Environment**. 2008c. Disponível em <http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_User_Guide_51519c.pdf>. Acessado em 10 Nov. 2010.

MICROCHIP. **PIC24FJ128GA010 Family Data Sheet. 2009.** Disponível em <http://ww1.microchip.com/downloads/en/devicedoc/39747D.pdf>. Acessado em 10 Fev. 2010.

SANTOS, J. C. S. **Projeto de um sistema microcontrolado utilizando Internet embarcada para monitoramento remoto em tempo real de temperatura e disponibilização dos dados na WEB através de conexão de rede**. Natal, RN. Trabalho de conclusão de curso. Universidade Federal do rio Grande do Norte. 2009.

Capítulo

16

O que sua personalidade revela?

Fidelizando clientes web através de Sistemas de Recomendação e traços de personalidade[17]

Maria Augusta S. N. Nunes, Silvio César Cazella

Abstract

Recently, studies from (Damásio 1994), (Simon 1983), (Picard 1997), (Trappl et al 2003), (Thagard 2006) and (Nunes 2009) have demonstrated how important psychological aspects of people such as personality traits and emotions are during the human decision-making process. Those studies have demonstrated how much subtle and inherent aspects from human personality have influenced the human interpersonal interaction. Indeed it proved how much those aspects could enhance the personalization during the human-human interaction and how it could be beneficial by offering products to customers in conventional business processes in the real world. Some studies (Reeves and Nass 1996) have been conducted showing that humans respond psychologically to computers and other media as if these were also human. Considering this aspect, no matter what kind of resource the computer use, the computer will be potentially making decisions and working with and for people. Thus, some understanding of the nature of human psychological aspects by computer is extremely relevant and necessary in order to improve its understanding. By improving those aspects, the computer could also improve the level of customization and optimization of their processes of decision making in order to enhance human-computer interaction and therefore the personalization on the web.

[17] Esse trabalho é uma reedição do minicurso apresentado no WEBMEDIA 2011. O nosso objetivo na reedição do curso é trazer a população regional o estado da arte do que esta sendo apresentado nos melhores eventos da área de CC do Brasil.

Resumo

Estudos recentes de psicólogos, neurologistas, antropólogos e cientistas computacionais (Damásio 1994), (Simon 1983), (Picard 1997), (Trappl et al 2003), (Thagard 2006) e (Nunes 2009) têm provado o quão importante os aspectos psicológicos humanos, tais como emoção e traços de personalidade, são no processo de tomada de decisão humana. Os mesmos estudos provam que esses aspectos sutis e inerentes a personalidade humana influenciam de maneira efetiva e particular suas interações interpessoais potencializando a personalização na interação humano-humano podendo substancialmente favorecer aspectos de processos comerciais convencionais na oferta de produtos e serviços no mundo real. Alguns estudos (Reeves and Nass 1996) têm sido conduzidos indicando que os humanos respondem psicologicamente a computadores e outras mídias como se esses fossem, também, humanos. Considerando esse aspecto, não importa que recurso o computador estará usando, entretanto, em todos os casos, o computador estará, potencialmente, tomando decisões e trabalhando com e para as pessoas. Assim, para o computador, o entendimento da natureza psicológica humana é extremamente relevante e necessária para que o mesmo possa melhorar sua compreensão do ser com quem interage, melhorando assim, o nível de personalização e otimização dos seus processos de tomada de decisão visando potencializar a interação humano-computador e conseqüentemente a personalização em ambientes web.

16.1. Introdução

Atualmente o uso da *web* como uma fonte de entretenimento, cultura, informação, de produtos e serviços é, de certa forma, indispensável às atividades diárias da grande parte das pessoas nas civilizações modernas. Atualmente a *web* é considerada pelas pessoas como uma fonte inesgotável de recursos, de todo tipo, onde tudo pode ser encontrado, executado, solucionado e, principalmente, onde tudo é possível e acessível. Nesse tipo de ambiente, onde os humanos podem virtualmente viver e usar qualquer recurso real/virtual para alcançar o desejado, a personalização das informações, produtos e serviços oferecidos aos usuários é fundamental. Não importa que tipo de recurso na *web* seja utilizado, em todos os casos o computador estará potencialmente trabalhando com, e, para as pessoas. Para que a personalização ocorra de forma adequada uma eficiente forma de realizar o processo de tomada de decisão computacional deve ser adotada.

Considerando a sobrecarga de informação disponibilizadas na *web* dificilmente a personalização de informações, produtos e serviços tem se dado de forma efetiva no Brasil. Na Europa e Estados Unidos esse problema tem sido contornado pelo uso efetivo de Sistemas de Recomendação que manipulam a grande massa de informação disponível na *web* filtrando o que realmente interessa ao usuário de *e-commerce* e *e-services*. Dessa forma, a *web* brasileira vem perdendo um grande potencial mercadológico pois o empresariado vem negligenciando esse aspecto. Há uma estranha contradição nessa questão, pois a Academia brasileira produz ciência e tecnologia suficiente para inovar as técnicas utilizadas em *e-commerce* e em *e-services* no Brasil, entretanto o conhecimento

produzido é subutilizado pelo empresariado brasileiro, o que acaba acarretando o déficit de nossa tecnologia comercial se comparado a Europa e Estados Unidos.

Assim, esse capitulo propõe diminuir o déficit disponibilizando um portfólio dos trabalhos no que tange tanto o estado da arte como o da técnica dos trabalhos em andamento que direcionam a área de Sistemas de Recomendação com o uso inovador da personalidade, subárea da Computação Afetiva, principalmente na linha de *e-commerce* e *e-services*.

A seguir apresenta-se a estrutura do capítulo: na seção 16.1 é discutido aspectos introdutórios do assunto proposto pelo capítulo bem como sua agenda; na seção 16.2 é introduzida a área Computação Afetiva, incluindo uma breve descrição dos aspectos que envolvem a afetividade enfatizando a personalidade; na seção 16.3, é descrito e exemplificado como, porque e quando a Computação Afetiva, principalmente a personalidade, potencializa a tomada de decisão humana. Apresenta-se, então, as abordagens de personalidade existentes exemplificando as abordagens codificáveis em computadores. Na seqüência, discute-se como os aspectos de personalidade influenciam na identidade do usuário e como isso afeta seu perfil. Seguindo-se pela discussão dos critérios de armazenamento existentes (*Markup Languages*, *Ontologies*, *User Profile*). Finalmente, são descritas as metodologias existem hoje para extração de personalidade por computadores; na quarta seção são descritas as formas existentes de tomada de decisão computacional enfatizando os Sistemas de Recomendação, focando em sua aplicação em *e-commerce*, *e-services* e TV digital. Questões relativas a técnicas, e estratégias de recomendação serão apresentadas com o resumo de seus algoritmos; a seção 16.5 apresenta a concatenação das duas seções anteriores. Nessa será exemplificado como Sistemas de Recomendação podem efetivamente personalizar ambientes com ganhos efetivos usando a personalidade; finalmente a seção 16.6 apresentada conclusões e perspectivas futuras da área de pesquisa, seguido pela seção 16.7 onde as referências bibliográficas são apresentadas.

16.2. Computação Afetiva

Desde a década de 70, cientistas computacionais, principalmente da área de Computação Afetiva buscam modelar e implementar aspectos psicológicos humanos em ambientes computacionais.

Na Computação Afetiva estuda-se como os computadores podem reconhecer, modelar e responder às emoções humanas (dentre outros aspectos) e, dessa forma, como podem expressá-las através de uma interface/interação computacional (Picard 1997). Acredita-se que permitindo que computadores expressem/captem fisiológica e verbalmente informações psico-afetivas, em uma interação humano-computador, é possível induzir e despertar afetividade em humanos. O principal objetivo de se promover esse interfaceamento afetivo é contribuir para o aumento da coerência, consistência, predicabilidade e credibilidade das reações personalizando as respostas computacionais providas durante a interação humana via interface humano-computador.

O usuário é um agente em um ambiente computacional (*web*, por exemplo) onde a interface, ou a adaptação da mesma, é fundamental para que se crie uma personalização da interação com o usuário contextualizado-o em sua zona de conforto e necessidades emergentes. Muitas características da identidade pessoal (aspectos psicológicos e habilidades sociais) do usuário podem ser captadas por meio de símbolos disponíveis na interface dos ambientes computacionais. Porém cada símbolo pode ser interpretado diferentemente por cada sujeito/usuário, considerando como o mesmo os projeta durante suas interações com o mundo real. Da mesma forma, os símbolos projetados via interface são, também, interpretados diferentemente por cada usuário, emergindo/brotando, dessa forma, diferentes aspectos psicológicos, tais como personalidade, durante sua interação no ambiente podendo influenciar definitivamente (positiva ou negativamente) na interação do usuário em ambientes computacionais atuais, tais como *e-commerce* e *e-services*.

16.3. Tomada de decisão humana & Computação Afetiva

Como descrito anteriormente, estudos recentes de psicólogos, neurologistas, antropólogos e cientistas computacionais (Damásio 1994), (Simon 1983), (Goleman 1995), (Paiva 2000), (Picard 1997), (Trappl et al 2003), (Thagard 2006) e (Nunes 2009) têm provado o quão importante os aspectos psicológicos humanos, tais como Emoção e personalidade, são no processo de tomada de decisão humano influenciando, assim, suas interações. Assim, para o computador, o entendimento da natureza psicológica humana é extremamente relevante e necessária para que se possa melhorar seu nível de personalização e otimizar a interação também em ambientes computacionais. Considerando essa necessidade, esse capítulo se propõe a tratar a questão da personalidade humana como forma de potencializar a interação humano-computador apresentando primeiramente abordagens psicológicas passiveis de codificação em computadores.

16.3.1. Personalidade

Na Psicologia não existe um consenso para a definição de personalidade. De acordo com Schultz (1990) a origem em latin da palavra personalidade "*Persona"* refere-se a mascara usada por um ator para a encenação de uma peça teatral ao público. Schultz ainda estende sua definição descrevendo personalidade como "um conjunto permanente e exclusivo de características identificáveis nas ações/interações do individuo em diferentes situações". Ainda, Burger (2000) define personalidade como "um padrão de comportamento consistente e processo intrapessoal que é originado internamente no indivíduo".

A personalidade é mais que apenas a aparência superficial e física de um indivíduo, ela é relativamente estável e previsível, porém ela não é necessariamente rígida e imutável. A personalidade, geralmente, permanece estável por um período de 45 anos iniciando na fase adulta (Soldz and Vaillant 1998). A personalidade pode ser definida segundo diversas abordagens, uma abordagem bastante interessante é a abordagem de traços de personalidade que permite diferenciar psicologicamente pessoas usando traços mesuráveis e conceituáveis. Traços de personalidade são formados por um conjunto de características humanas factíveis de modelagem e implementação em computadores (Nunes 2009).

Os traços de personalidade foram historicamente definidos por Allport (1927). Allport criou 17.953 traços (“comuns” e “individuais”) para descrever a personalidade de um indivíduo. Logo após Allport, pesquisadores assumiram que todos os homens eram identificáveis “como algum outro homem” e, dessa forma, a maioria das diferenças individuais (representadas pelos traços individuais e Allport) eram insignificantes nas interações diárias humanas e, assim eles limitaram exponencialmente o número de definições de traços. Posteriormente, os pesquisadores reduziram mais de 99% dos traços. Acabaram restando cinco fatores que se replicaram em seus estudos empíricos, como resultado, o modelo *Big Five* (John *and* Srivastava 1999) foi criado. Porém, mesmo considerando que o *Big Five* representasse grande eficiência na representação da estrutura de personalidade, ele não garantia exaustivamente todas as dimensões de personalidade. Dessa forma, *facetas* também foram criadas e usadas pelos psicólogos para dotar o *Big Five* de características mais detalhadas (Goldberg *et al* 2006).

Para que exista possibilidade de se personalizar a oferta e recomendar produtos, serviços em um ambiente *web* à um usuário é necessário ter-se conhecimento sobre quem é este usuário. Antes mesmo de pensar em capturar e armazenar a informações pessoais e comportamentais dele é necessário identificar que o tipo de informação será relevante para a geração da personalização adequada. Para a correta geração da recomendação a definição do perfil do usuário e tipo de informações usada é imprescindível. Abaixo apresenta-se como a personalidade influencia nessas definições através de sua influência na identidade e, conseqüentemente no o perfil do usuário.

16.3.2. Identidade e Perfil de Usuário

16.3.2.1 Identidade do Usuário

Segundo a visão da psicologia clássica, identidade é definida pela autoconsciência/visão que cada pessoa possui de si mesma, enquanto que na Psicologia Social e Sociologia, identidade pode ser definida como a forma que cada pessoa é vista sob os olhos da sociedade.

Segundo os pesquisadores de Teoria da personalidade, o desenvolvimento da identidade recebe uma importante influência da personalidade. Boyd (Boyd 2002) descreve dois aspectos diferentes da identidade: a noção internalizada do “eu” (identidade interna) e a versão projetada da internalização do “eu” (identidade social). Nessa mesma linha, Erikson (1980) por exemplo, acredita que identidade (EGO) tem uma representação pessoal interna (identidade interna) bem como uma representação social (identidade social). Giddens (1991) concorda que sem experiências sociais o “eu” não pode internalizar evolução. Giddens ainda afirma que a identidade de um individuo não é estática, ela pode ser representada em constante evolução, principalmente porque o componente social é dinâmico e esta sempre sendo modificado. Mead (1934) ainda define “eu” e “mim”, onde “mim” representa o aspecto socializado da pessoa (identidade social), enquanto que o “eu” representa como a pessoa se define em relação aos outras pessoas da sociedade (identidade individual).

Note que no mundo virtual onde não há presença física e conseqüentemente não há percepção de características sutis da identidade, várias pistas que possivelmente identificariam dicas de preferências, comportamentos, habilidades sociais, entre outras, são ausentes, ao contrário do que ocorre no mundo real (Donath 1999). Donath (2000) afirma que conhecer a identidade da pessoa é vital para uma adequada personalização de uma ambiente no mundo virtual, como na *web* por exemplo. Goffman (1959) afirma ainda, que as pessoas se esforçam para se apresentar como "aceitáveis" aos olhos da sociedade (em comunidades virtuais, por exemplo).

Considerando a identidade como um canal importante onde as características objetivas e subjetivas das pessoas emergem, denomina-se de fundamental importância seu uso em Sistemas de Recomendação no intuito de fornecer pistas sobre os futuros comportamentos e necessidades dos usuários em um dado ambiente onde a personalização se faz eficaz.

Tecnicamente, em Ciência da Computação, a tecnologia usada para formalizar a identidade em um dado ambiente computacional é pelo uso de Perfil/Modelo do Usuário (identidade Interna) e Reputação do Usuário *(*identidade Social).

16.3.2.2 Perfil de Usuário

Donath (1999) afirma que para a formação eficiente de uma identidade Virtual é crucial que o usuário tenha definida sua identidade interna e sua identidade social. No mundo virtual a identidade interna do usuário é definida por ele próprio similar ao mundo real (algumas vezes também é descoberta através de técnicas de *Machine Learning*). Enquanto a identidade social é definida pelos outros membros do mundo virtual (elucidada na próxima seção). Tanto a identidade interna, como a identidade social são armazenadas no perfil do usuário.

Perfis de usuários são conceitos aproximados, eles refletem o interesse do usuário com relação a vários assuntos em um momento particular. Cada termo que um perfil de usuário expressa é, num certo grau, características de um usuário particular (Poo et al 2003) incluindo todas as informações diretamente solicitadas a ele e aprendidas implicitamente durante sua interação na *web* (Carreira et al 2004). Fisicamente, o perfil do usuário pode ser visto como uma base de dados onde a informação sobre o usuário, incluindo seus interesses e preferências, é armazenada e pode ser dinamicamente mantido (Rousseau et al. 2004), (Poo et al. 2003).

Na *web* encontram-se muitos tipos de perfis de usuário com diferentes graus de complexidade. Eles são desenvolvidos no contexto de *e-commerce*, *e-learning* e *e-community*, por exemplo. Kobsa (2007) cria uma modelagem genérica de usuário para ser usada como uma *shell* para a criação de categorias de informação sobre o usuário objetivando personalizar as aplicações *web*. O modelo proposto por Kobsa é um dos mais reputados. Paiva e Self (1995) também desenvolveram uma *shell* de modelo de usuário chamado *TAGUS*, criado para melhor modelar os alunos para atividades de aprendizado. No *e-commerce*, (Riedl et al. 1999), (Herlocker et al. 2004), (Konstan et al. 1997), (Schafer

et al. 1999) e (Schafer et al. 2001), do GroupLens, criaram vários modelos de usuário baseado em rankeamento de filmes, de notícias, entre outros. Esses modelos têm sido usados nos Sistemas de Recomendação criados pelo GroupLens.

Considerando ainda definições de modelo de usuário, Heckmann (2005) e Heckmann e Kruguer (2003) propõem uma ontologia[18] de um modelo de usuário geral *(GUMO). O GUMO* é um modelo ubíquo de modelo de usuário incluindo muitos aspectos básicos de usuário, partindo desde informação de contato, demográficos, habilidades fisiológicas e psicológicas, estado emocional, estado mental e nutrição. A ontologia de Heckmann é muito rica e pode ser implementada de acordo com o interesse do projetista de uma *shell* de perfil de usuário.

Note que para gerar as recomendações e personalizar o ambiente ao usuário, os Sistemas de Recomendação necessitam da identidade interna do usuário que é definida pelo perfil de usuário, como foi apresentado. Porém necessita, muitas vezes, também, da identidade social que é definida pela reputação do usuário.

16.3.2.3 Reputação

Reputação pode ser definida como o retorno social recebido sobre a personalidade de alguém. A reputação pode ser compatível ou não com a descrição feita no perfil de usuário. Josang et al (2007) descreve reputação como "a informação normalmente dita ou crível sobre as características de uma pessoa ou coisa e seus estados".

Resnick et al. (2000) define reputação como a coleção dos *feedbacks* recebidos sobre o comportamento efetuado pelos participantes de uma comunidade. A reputação ajuda as pessoas a escolher parceiros confiáveis no mundo virtual que são credíveis no mundo real. Geralmente nas redes de reputação, os usuários encorajam os comportamentos confiáveis discriminando a participação de pessoas desabilitadas moralmente ou desonestas.

Segundo Rein (2005) a reputação pode ser também definida como um completo sistema de informações de opiniões alheias sobre um usuário, que inclui todos os aspectos de um modelo de referência. Esse modelo de referência é baseado em nove aspectos determinantes: conhecimento, experiência, credenciais, endosso, contribuições, conexões, sinais, *feedback*, contexto e valores sociais. A visão estrutural de Rein descreve as funcionalidades e comportamentos essenciais do ser humano que são desejáveis e efetivos para possivelmente ser representado através de uma reputação explicita e fácil de ser medida no usuário.

A reputação é geralmente aplicada para gerenciar comportamento do usuário durante um processo comercial (*e-commerce*, por exemplo) envolvendo compra e venda de produtos e/ou serviços e também durante processos sociais como combinação social em comunidades virtuais e redes sociais.

Em processos comerciais, como por exemplo, no *eBay* (Resnick et al. 2000)

[18]Uma ontologia é uma especificação de um conceito.

(Resnick et al. 2006) um consumidor compra um certo produto de alguém. Depois disso, ele deixa um *feedback* sobre o produto comprado e/ou o comportamento do vendedor durante o processo de venda.

Em contraste, em situações sociais como, por exemplo, *Orkut, IKarma, Opinity, LinkedIn, Mendeley* (Jensen et al. 2002), usuários são membros de comunidades virtuais ou redes sociais. Eles são capazes de coletar gerenciar e promover reputação de usuário entre seus clientes e contatos da comunidade ou rede. Isto é, usuários (prestadores de serviço) que tem *profile* na rede de reputação, que é também uma rede social podem ser "*tagged*" e rankeados pelos seus clientes e/ou contatos. Usuários podem ser encontrados através de *tags* em e-mail ou, também, alguém pode encontrar um contato de um prestador de serviço simplesmente procurando em *tags* na própria rede de reputação.

Nunes (2009) convenciona reputação como uma extensão de um perfil de usuário. Teoricamente se usa o mesmo tipo de informação armazenada no perfil de usuário, porém o conjunto de informações é fornecido por outro alguém (amigo, cliente do usuário, entre outros). Nesse caso, a identidade é determinada pelos traços de personalidade do usuário informados (ou automaticamente medidos) por ele mesmo para perfil de usuário e informados (ou automaticamente medidos) por uma outra pessoa para reputação de usuário.

O perfil de usuário é muito importante para definir a identidade do usuário. Dessa forma o perfil pode prever necessidades e comportamentos do usuário em um ambiente computacional, enquanto a reputação permite a criação de relação de confiança entre membros de uma comunidade em um ambiente *web*, especialmente em *e-commerce*. A identidade do usuário é muito útil para sua interação social no ambiente computacional.

A seguir, apresentam-se os esforços dos cientistas de Ciência da Computação e Computação Afetiva com o objetivo de padronizar uma forma de armazenamento e padronização da informação afetiva já definida no perfil do usuário e/ou reputação.

16.3.3. Mecanismos para armazenamento da personalidade em computadores

Considerando a grande gama despadronizada de representação afetiva, alguns pesquisadores pertencentes ao W3C Emotion Incubator Group (W3C 2010a) têm ampliado esforços para definição de uma padronização formal da afetividade através de uma linguagem de marcação. A padronização afetiva tem focado, principalmente, na representação de emoções em uma *markup language*, considerando que emoção é o campo da Computação Afetiva historicamente mais desenvolvido. Após esforços iniciais de Heckmann (2005) na criação de uma ontologia completa incluindo diversos aspectos de Personalidade, a partir de 2008, alguns esforços pontuais de pesquisadores, tais como Nunes (2008; 2009; 2010b) tem seguido na direção duma padronização especifica modelada através de uma *markup language* intitulada de PersonalityML.

16.3.3.1 XML & Markup Languages

Uma linguagem de marcação é uma forma de definir e identificar a estrutura e o significado do conteúdo em um documento. A especificação XML define um padrão para adicionar marcação a documentos, genericamente, o XML nem define semântica nem um conjunto de elementos pré-definidos, como o HTML. Na realidade, a XML pode ser considerada como uma meta-linguagem para definir linguagens de marcação, fornecendo mecanismos (através de sua especificação) para que se defina elementos e relacionamentos entre eles, para semânticas preconcebidas.

O uso do padrão XML (W3C 2010d) para definição de outras linguagens de marcação é algo amplamente desenvolvido pela comunidade acadêmica (e são extensamente utilizadas comercialmente). Alguns casos fazem parte da própria tecnologia XML, por exemplo, XML Schema e XSLT. No primeiro caso, o padrão XML Schema possui elementos pré-definidos que definem como construir esquemas para instâncias de documentos XML. No caso da XSLT, os elementos definem como um processador deve ler uma instância XML e transformá-la para outro formato texto como, por exemplo, um HTML, um documento em LaTeX ou até mesmo um simples txt. Outros exemplos são: Simple Object Access Protocol - SOAP (W3C 2010b), *Attention Profiling Markup Language* - APML (Angell et al 2010), Extensible MultiModal Annotation markup language - EMMA (W3C 2010c), Emotion Markup Language - EmotionML(W3C 2010a) entre diversas outras[19].

16.3.3.2 EmotionML & PersonalityML

A EmotionML (*Emotion Markup Language*) versão 1.0 foi apresentada pelo W3C (*WWW Consortium*) em outubro de 2009. Concebida como uma extensão da linguagem de marcação XML, em uma primeira tentativa da comunidade científica para padronizar a representação de emoções. A EmotionML provê elementos para uma linguagem de marcação que chama a atenção ao equilibrar a fundamentação científica e a aplicação prática. A EmotionML é uma linguagem dinâmica, ainda que muito jovem e com uma representação ainda bastante genérica. Apesar de flexível, a EmotionML não leva em consideração outros aspectos que afetam significativamente a correta identificação de uma emoção, como por exemplo a Personalidade.

Na realidade, apesar de existência de alguns avanços significativos quanto à Personalidade, são relativamente poucos e incipientes os trabalhos que lidam computacionalmente com este aspecto psicológico, e menos ainda aqueles que lidam com seu reconhecimento automático ou mesmo representação computacional (Mairesse et al 2007), (Heckmann 2005), (Nunes 2009), (Nunes et al 2010).

Como descrito anteriormente, pesquisadores em Computação Afetiva têm implementado as emoções explicitamente, mas de forma despadronizada antes do surgimento o da EmotionML. A representação da emoção, ao invés de outros aspectos de Computação Afetiva se tornou possível devido às emoções serem mais facilmente

[19] Consultar http://en.wikipedia.org/wiki/List_of_XML_markup_languages para uma lista não exaustiva.

mensuráveis e interpretáveis e poder efetivamente influenciar diretamente na ação-interação dos usuários. As emoções são instantâneas, elas têm uma vida curta, volátil e mudam constantemente, diferentemente da personalidade que é um estado muito mais estável e, normalmente, mantido durante um período de 45 anos. Apesar disso, com base nas pesquisas é possível dizer que a personalidade implica em emoções (Lisetti 2002); cada pessoa ou agente que tem emoções tem uma personalidade; e, geralmente, a personalidade não aparece explicitamente mesmo que influencie as emoções diretamente. Lisetti (2002) descreve um modelo complexo para representar aspectos psicológicos em agentes inteligentes (virtual/real) que interagem socialmente, denominado *Affective Knowledge Representation* (AKR - Representação do Conhecimento Afetivo). No AKR apresenta-se a Personalidade como o topo do modelo hierárquico dos aspectos psicológicos, denotando assim seu maior poder.

Dessa forma, considerando que a personalidade é mais abrangente e implica na emoção, Nunes et al (2010b) propõem uma extensão à representação padronizada de emoção incorporando uma nova proposta intitulada de PersonalityML, que é também baseada em XML. Essa extensão objetiva representar a complexidade afetiva descrita brevemente em (Nunes et al 2010b), onde a personalidade é o ponto chave da cadeia e, que sua representação está limitada pela atual versão da EmotionML.

Entretanto, antes de se pensar em uma linguagem de marcação para a personalidade, deve-se lembrar que nem todas as teorias de Personalidade existentes possuem uma estrutura passível de representação e, consequentemente, de implementação em computadores. Felizmente há algumas que possuem, sendo as mais utilizadas na literatura da Computação Afetiva, àquelas pertencentes à abordagem dos Traços, teoria esta que serviu de base para o lançamento da primeira versão da PersonalityML (Bezerra et al 2011) sendo modular suficiente para a incorporação automática de novas teorias e abordagens existentes hoje. Dessa forma, após a formalização e criação de mecanismos de armazenamento, abaixo apresenta-se como a personalidade pode ser extraída do usuário.

16.3.4. Extração de personalidade

Considerando a abordagem de traços, escolhida por ser a melhor forma para representação de personalidade em computadores, psicólogos geralmente usam questionários intitulados de inventários de personalidade. Esses inventários são diretamente aplicados por psicólogos ou encontrados na *web*. Os mesmos podem ter uma pequena ou grande quantidade de questões e o número de questões é diretamente proporcional a granularidade e precisão dos traços de personalidade extraídos do usuário. Segundo Gosling (2008), os longos e mais precisos inventários tomam um tempo bastante considerável do usuário e muitas vezes torna inviável sua aplicação, nesses casos opta-se pelo uso de testes compactos como o TIPI (estando ciente da limitação quanto a precisão das respostas) ou mesmo opta-se pelo uso de outras formas de extração de personalidade, muitas vezes em estágio embrionário de desenvolvimento. Inicialmente apresenta-se ao leitor a forma clássica de extração de personalidade seguindo-se por abordagens embrionárias e direções de pesquisa:

16.3.4.1 Extração de Personalidade através de inventario baseado em Traços

Existem diversos inventários validados, como apresentado em (Nunes 2009; 2010). Porém um Teste de personalidade bastante interessante é o NEO-IPIP (Johnson 2000), (Johnson 2005) desenvolvido em conjunto com o *International Personality Item Pool* (Goldberg *et al* 2006). Ele permite medir as cinco dimensões do Big Five incluindo mais seis facetas para cada dimensão (30 facetas no total) usando uma descrição detalhada dos traços de personalidade humana e por conseqüência propiciando uma grande precisão na representação da personalidade.

Segundo Gosling (2008) "deixamos pistas sobre nossa personalidade em tudo o que fazemos, em nossos objetos, onde vivemos". Sendo assim, através dos padrões de navegação de cada usuário, é possível adquirir características psicológicas, através de uma abordagem implícita e transparente ao usuário. Um pensamento natural é que esta seria a melhor forma de obtenção dos traços de personalidade dos usuários, uma vez que esta abordagem exige menor esforço cognitivo se comparado aos tradicionais inventários de personalidade.Porém, Dumm et al (2009) negaram esta hipótese ao pesquisar três interfaces de obtenção de traços de personalidade, sendo duas delas explícitas e uma implícita. O resultado foi que a interface NEO (utilizada nos testes como NEO-PI-R e NEO-IPIP) obteve melhor *feedback*, tanto em termos de resultados apresentados quanto em termos de facilidade de uso. Dessa forma, o teste NEO-IPIP, citado acima, torna-se, então, uma opção interessante a ser utilizada como ferramenta de entrada de dados explícita para obtenção dos traços de personalidade dos usuários. Este teste possui um *feedback* positivo em mais de 99% dos casos (Johnson 2005).

Assim, a equipe da autora propôs uma nova interface ao NEO-IPIP, intitulada *Personality Inventory*, oferecendo mais usabilidade ao ambiente proposto originalmente por Johnson (2000) e Gosling (2008). As medições de personalidade extraídas no inventário seguem os padrões propostos por Johnson (2000) e Gosling (2008). Para que os resultados pudessem ser mantidos em um local seguro e persistente e serem usados em diversas aplicações de pesquisa, optou-se por uma aplicação *web*. A interface é simples como apresentado na Figura 16.1. Os inventários disponíveis estão apresentados na Figura 16.2.

Figura (16.1) Interface inicial (1b) Inventários Disponíveis (1c) Questão 1 do inventário

Como visto na figura 1a, para que a medição de personalidade seja feita, o usuário precisa criar uma conta de usuário e responder pelo menos um dos questionários disponíveis, no caso, o NEO-IPIP, como apresentado na figura 1b. Após a seleção do Inventário NEO-IPIP o usuário deve responder o questionário como apresentado na figura 1c.

Uma vez respondido o questionário do NEO-IPIP, cada questão respondida pelo usuário tem valor atribuído entre 1-5. Ao finalizar o teste, os valores atribuídos a cada uma das questões respondidas são utilizados para calcular o resultado. No cálculo, o resultado é normalizado e é atribuído um valor entre 1-100 para cada um dos itens do Big Five, bem como para as suas facetas. Note que somente então o relatório descritivo dos traços de personalidade do usuário é gerado e disponibilizado exclusivamente a ele. Na Figura 2, parte do relatório descritivo dos traços de personalidade do usuário, extraído do inventário NEO-IPIP, é apresentado.

Uma vez visualizado seu prognóstico, o usuário poderá re-visualizar o resultado através da própria aplicação ou exportá-lo como PersonalityML.

16.3.4.2 Extração de Personalidade através de outras técnicas

Note que Gosling (2008) afirma que a melhor forma de obtenção dos traços de personalidade dos usuários é através do uso de uma abordagem que não exija esforço cognitivo se comparado aos tradicionais inventários de personalidade, como o exemplo mostrado acima. Andrade et al (2011) afirma que os traços medidos através de inventários de personalidade muitas vezes são, em parte, um conjunto de dados provenientes do auto-relato da própria opinião do usuários podendo desvirtuar da sua real personalidade.

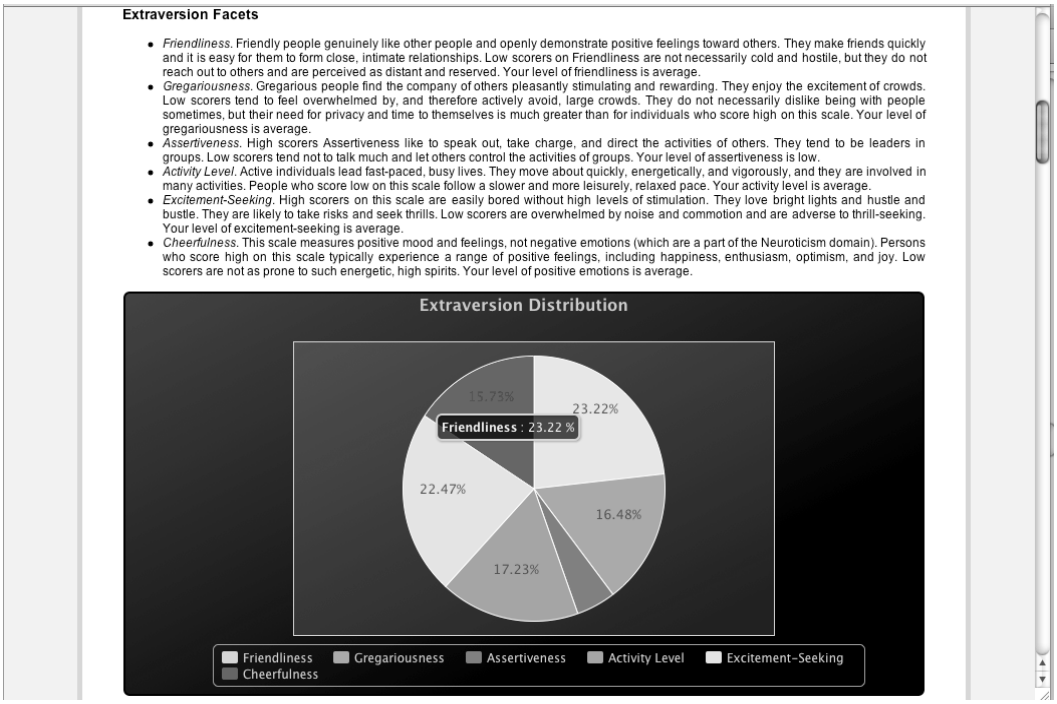


Figura 16.2. Relatório descritivo parcial do NEO-IPIP (Nunes 2010)

Porém, infelizmente, em computadores, ainda poucas técnicas de extração de traços de personalidade, que diferem dos tradicionais inventários, tem sido desenvolvidas e implementadas. Segundo Brinkman e Fine (2008) constantemente pesquisadores tem se aventurado em novas pesquisas que tentam obter personalidade de uma forma menos intrusiva que os tradicionais inventários. Segundo Porto et al (2011) tem-se realizado tentativas através da análise de gravações automáticas de dados da interação do usuário com o sistema, tais como a avaliação a escolha dos usuários para customização de algumas aplicações como *Windows Media Player, ICQPlus* e o *Winamp*, por exemplo.

Andrade e Nunes (2011) afirmam ainda, que nesse novo contexto, incluem-se as técnicas de Reconhecimento de Padrões, baseadas em cálculos probabilísticos visando reconhecer emoções ou mudanças de comportamento através da observação de um usuário utilizando o determinado sistema em uso no computador, seja pelo clique do *mouse*, por teclagem, captura de imagem do usuário pela *webcam* ou, ainda, por sensores que capturam sinais vitais dos usuários (as principais referências são os trabalhos de (Chanel 2009), (Hu and Pu 2009), (Tkalčič 2010) e (Khan et al 2008)). Rabelo e Nunes (2011) ainda afirmam que é possível identificar traços de personalidades a partir de definições de modelos e Frameworks que utilizam recursos estatísticos para classificar determinado conjunto de dados, definido com conjunto de características, como pertencentes a um determinado traço de personalidade, associada a classes no reconhecimento de padrões. Autores como Mairesse e Walker (2005; 2008); Hussain e Calvo (2009) ; Pianesi et al (2008) convergiram as suas técnicas e modelos à identificação de personalidade a partir de extratos de diálogos de diversas modalidades, tais como: comunicação textual assíncrona (*e-mails* e dialogo em sites de relacionamento), conversa falada (áudio presencial ou virtual), expressões faciais (videoconferência), sinais fisiológicos, dentre outros. Ainda Gill e Oberlander (2003) alegam a possibilidade de detecção de traços de personalidade em textos de comunicação assíncrona, mas explicitamente as mensagens de correio eletrônico através de processos estatísticos.

Abaixo, de acordo com, Rabelo e Nunes (2011) e Porto et al (2011) alguns trabalhos que envolvem extração de traços de personalidade, são melhor detalhados:

- Em Porto et al (2011), a extração de traços de personalidade é realizada através do teclado é feita obtendo-se informações de como o usuário digita determinado texto, essa informação pode ser obtida através da latência entre teclas consecutivas (o tempo entre o apertar de uma tecla e da subseqüente). Esse tempo é obtido através de um evento de teclado, o *keyPress*. O evento aciona um *timer* que é interrompido quando o evento é acionado novamente, ou seja, o *timer* conta o tempo decorrido entre uma tecla pressionada e outra, o tempo de latência. Tendo a informação de como um usuário digita, este pode ser comparado com um banco de dados que contem informações de outros usuários, informações estas que são o ritmo de digitação e a personalidade do usuário, e através de técnicas de agrupamento de dados (clustering), o usuário é isolado em um grupo que tem o mesmo padrão de digitação, e então os traços de sua personalidade são inferidas a partir dos usuários que fazem parte do mesmo grupo. Os testes realizados para essa abordagem tendem a confirmar que quando os usuários são agrupados levando em consideração o ritmo de digitação, alguns traços de

personalidade são correlacionados. Usuários que são agrupados num mesmo cluster tendem a ter o mesmo valor para algumas facetas, e, portanto usuários que não se soubessem o valor das facetas, tendo sido agrupados em um cluster, poderiam ter seus valores inferidos pelos outros elementos do mesmo cluster.

- Gill e Oberlander (2003) investigam o impacto da iteração mediada por computador na percepção pessoal, em particular as características importantes para a socialização e colaboração, Extroversão e Neuroticismo, determinando que tais características podem ser detectadas a partir do texto de um e-mail e para a determinação das características linguísticas da personalidade destes, um conjunto de analises e técnicas foram aplicadas: Os dados LIWC(Linguistic Inquiry Word Count), base de dados e propriedades psicolinguísticas de derivados do *Medical Research Council*, além da medida de diversidade léxica conhecida com *Type Token Ratio* (TTR).

- Por sua vez, Mairesse e Walker (2005) propõem em seu trabalho uma modelagem para reconhecimento de personalidade em conversação, intitulado de *Personality Recognizer*, a partir de uso de modelos estatísticos não lineares para classificação baseada em traços de personalidade do Big Five. A abordagem pode ser resumida em cinco etapas: (1) Recolher individual dos Corpora (Textos); (2) Coletar informações de personalidade para cada participante; (3) Extrair características relevantes a partir dos textos; (4) Construir modelos estatísticos das avaliações de personalidade com base nas características; (5) Testar os modelos aprendidos nas saídas lingüísticas dos indivíduos invisíveis. Os dados consistem extratos de conversas diárias de 96 participantes, utilizando um gravador ativado eletronicamente durante o período de dois dias. Para preservar a privacidade dos participantes, bits aleatórios de conversa foram registrados, e somente as afirmações dos participantes foram transcritas, tornando-se impossível reconstruir conversas inteiras. O corpus contém 97.468 palavras e 15.269 declarações e o experimento foi dado através das etapas seguinte: (1) Seleção de Recursos: Recursos são automaticamente extraídos de cada trecho e é calculada a proporção de palavras em cada categoria a partir da ferramenta LIWC (Linguistic Inquiry Word Count), correlacionando ao Big Five. Estas características psicológicas adicionais foram computados pela média de contagens de palavras características de um banco de dados psicolinguística MRC; (2) Modelos Estatísticos: O treinamento dos modelos foi feito utilizando o algoritmo RankBoost, que classificar de acordo com cada característica do Big Five expressando a aprendizagem dos modelos como regras, que suportam a análise das diferenças nos modelos de personalidade, onde para cada extrato da conversa, a regra modifica a pontuação e assim o ranking. Os resultados obtidos são os aspectos caracterizam a produção de linguagem: (1) Tipos de expressão, conteúdo e sintaxe (LIWC); (2) As estatísticas psicolinguísticas (MRC); (3) Prosódia. Para avaliar como cada conjunto de recursos contribui para o resultado final, foram treinados os modelos com o conjunto completo de recursos e com cada conjunto, individualmente e testes T pareado mostram que os modelos de extroversão, afabilidade, consciência e abertura ao novo usando todos os recursos são melhores do que a linha de base aleatória. O Neuroticismo é o traço mais difícil de modelar, mas é melhor predito por características MRC. As características LIWC têm desempenho significativamente melhor do que a linha de base para todas as dimensões. As características prosódicas são boas preditoras de extroversão.

- Finalmente no trabalho estudado, Pianesi et al (2008) propõe um modelo para reconhecimento multimodal de traços de personalidade em interações sociais, que consiste em criar classificadores capazes de predizer os traços de personalidade, considerando o comportamento de um sujeito em um minuto de uma janela temporal. Como classificador, foi utilizado Maquinas de Vetor de Suporte (SVM), que tenta encontrar um hiperplano que não só discrimina as classes, mas também maximiza a margem entre elas. Como experimento, pressupõe-se que a personalidade aparece no comportamento social e é esperado que características audiovisuais fossem apropriadas para constituir um sistema automático de exploração e classificação de traços de personalidade. Duas abordagens são descritas: Na primeira abordagem, leva-se em consideração apenas o comportamento do sujeito é suficiente. Ex.: Maneira como se move entonação da voz e etc. Na segunda leva-se em consideração além do comportamento do sujeito, também o contexto social. Foram testadas as seguintes dimensões: Características Acústicas e Características Visuais e divididas em Níveis distintos: ALL(Todas as características Acústicas + Todas as Características Visuais), SEL (Seleções de características Acústicas + Seleções de Características Visuais) e No-Feat (Todas as características Acústicas + Características Visuais ; Características Acústicas + Todas as Características Visuais, etc). A Combinação destes níveis , exemplo (ALL , No-Feat) corresponde a um vetor de características utilizadas para treinar e testar os Classificadores que serão responsáveis por definir as distribuições dos dois traços de personalidade determinados como classes: A Extroversão e O Locus de Controle. Foram comparados os resultados das classificações com um classificador trivial que atribui a classe mais frequente para cada instancia. Os resultados do experimento determinaram que, duas analises de Variância, uma para cada tipo de personalidade mostram que os principais efeitos são significativos para $p < 0,0001$. Os efeitos para Extroversão não foram significativos em $p > 0,05$, enquanto os efeitos para Locus de Controle foram significativos em $p < 0,05$.

16.4. Tomada de decisão computacional & Sistemas de recomendação

16.4.1. Sistemas de Recomendação

Segundo Resnick e Varian (1997) muitas vezes é necessário fazer escolhas sem um grande conhecimento das possibilidades que nos são apresentadas. Normalmente nos baseamos nas recomendações de amigos, através de opiniões de especialistas ou, ainda, a partir de guias. Os Sistemas de Recomendação ajudam e aumentam este processo social natural já existente entre as pessoas.

Estes sistemas se tornaram uma importante área de pesquisa com os primeiros trabalhos sobre filtragem colaborativa e filtragem baseada em conteúdo em meados da década de 90 (Adomavicius e Tuzhilin, 2005). O interesse nesta área continua grande por ela ser rica em problemas de pesquisa e pela quantidade de aplicativos desenvolvidos para ajudar os usuários a lidar com a imensa demanda de informação, gerando recomendações personalizadas, conteúdos e serviços para os mesmos.

De acordo com Cazella et al (2010), as pessoas têm acesso a uma vasta gama de informações devido à grande oferta e aos recursos da Internet, porém despendem muito tempo na busca do que realmente é interessante ou útil para elas. A dificuldade de

encontrar a informação correta é aumentada quando a informação disputa a atenção de uma pessoa com uma série de outras informações não tão relevantes.

Resnick e Varian (1997) afirmam ainda que um sistema típico agrega e direciona as avaliações de itens feitas pelos usuários e as disponibiliza como recomendações para os indivíduos considerados potenciais e interessados neste tipo de recomendação. O grande desafio está em descobrir o relacionamento de interesses entre os usuários, realizando desta forma o correto casamento entre os itens que estão sendo avaliados e os usuários que estão recebendo a referida recomendação. O sistema gerará respostas mais valiosas quando as preferências dos usuários diferirem uma das outras do que quando temos uns poucos especialistas.

16.4.2. Técnicas de recomendação

Ao contrário da recuperação de informação, onde o usuário necessita informar sua necessidade de informação e é o responsável pelo início da interação, as técnicas de filtragem buscam informações relevantes geralmente através de um perfil de interesses do usuário. A seguir serão descritas algumas técnicas de recomendação (filtragem de informação) aplicadas a Sistemas de Recomendação.

16.4.2.1. Filtragem Baseada em Conteúdo

A filtragem baseada em conteúdo (FBC) realiza uma seleção baseada no conteúdo dos itens (Adomavicius e Tuzhilin 2005). Seu objetivo é gerar de forma automatizada descrições sobre o conteúdo dos itens e compará-las com os interesses do usuário (Herlocker 2000; Ansari 2000). Através disso, pode-se verificar qual a correlação entre o que o usuário está buscando e o item apresentado (Balabanovic e Shoham 1997).

A limitação da técnica está diretamente interligada com os objetos que este sistema necessita recomendar. Portanto, é necessário que o conteúdo dos itens seja computado automaticamente ou, então, será preciso que estas informações sejam adicionadas manualmente, como por exemplo, com a associação de descritores (*tags*). Enquanto existem técnicas que funcionam bem com o processamento de documentos de texto, alguns outros domínios têm problemas com a extração automática. Exemplos são: informação multimídia, imagens, áudio e vídeo (Ansari 2000).

Um grande desafio nessa área é o tratamento de metadados. A forma de identificar o conteúdo dos atributos e conseguir inferir que, por exemplo, “gripe” tem o mesmo significado de “resfriado” em determinado contexto. Para solucionar este problema, tem sido amplamente utilizada a técnica de indexação de frequência de termos (*term frequency indexing*) (Herlocker 2000).

Neste tipo de indexação, as informações sobre o item e as necessidades do usuário são descritas por vetores com uma dimensão para cada palavra da base de dados. Cada componente do vetor representa as repetições da respectiva palavra no item. Assim, os metadados serão ordenados por frequência e quanto mais próximos estiverem ao índice, mais relevante o mesmo será. Obviamente, métodos de pré-processamento de texto são

necessários para que palavras muito usadas em notícias e artigos (tais como preposições, artigos e conjunções) não sejam as palavras com maior frequência, prejudicando a predição de conteúdo.

Um problema que pode ocorrer é não conseguir distinguir dois itens caso sejam representados pelo mesmo grupo de características. Portanto, se dois artigos forem representados pelas mesmas palavras, não será possível mensurar a qualidade dos documentos (Ansari 2000).

Os usuários têm a liberdade de poder mudar de opinião e de hábitos. Sistemas de recomendação usando FBC podem tornar-se superespecializados quando o sistema já aprendeu demais sobre os hábitos daquele usuário, e não conseguirem sugerir coisas que fujam desse padrão. Um bom exemplo é o caso de um usuário que consome carne e que se tornou vegetariano. Não importa quanto tempo passe, o sistema vai continuar indicando carnes para compra. Este problema também não possibilita ao usuário receber recomendações de itens que ele já tenha gostado. Desta forma, uma pessoa que nunca tenha experimentado cozinha grega, jamais irá receber a sugestão de visitar o melhor restaurante grego da cidade (Ansari 2000).

Segundo Adomavicius e Tuzhilin (2005), uma forma de tentar resolver o problema da superespecialização é sugerir, aleatoriamente, alguns itens ao usuário. Adicionalmente, deve-se prestar atenção nos itens que são muito similares aos anteriormente vistos, pois o usuário deve receber recomendações homogêneas. Por exemplo, não é necessariamente uma boa idéia recomendar todos os filmes de Woody Allen para alguém que tenha gostado de um deles.

O problema do novo usuário também está presente nesta técnica. Acontece quando o sistema não consegue entender as preferências do mesmo, pelo fato de não ter avaliado uma quantidade mínima de itens, não gerando nenhuma recomendação. E, ainda, caso o usuário não possua muitas recomendações, a qualidade destas também pode ser baixa.

16.4.2.2. Filtragem Colaborativa

Para Cazella (2006), a essência dos sistemas colaborativos está na troca de experiências entre as pessoas que possuem interesses comuns. Para tanto, ao invés do conteúdo dos itens, a filtragem é baseada nas avaliações feitas pelos usuários daqueles itens.

A filtragem colaborativa (FC) foi proposta para suprir as necessidades da filtragem baseada em conteúdo (Herlocker 2000). Desta forma, os usuários receberão recomendações de pessoas com gostos similares e que gostaram do item no passado, enquanto que na FBC serão recomendados itens similares aos quais o usuário gostou no passado. É um processo que busca usuários similares e calcula, baseado na semelhança entre os mesmos, qual a nota aproximada que o usuário em questão daria ao produto caso o fizesse. Esta técnica também é chamada de "*k-nearest-neighbor*" ou "*user-based*" (Herlocker 2000).

Um ponto fraco da FC é o seu desempenho. Ela tem de ser rodada *online*, e caso seja executada para todos os usuários, leva-nos a um desempenho de $\Omega(N^2)$ (limite assintótico inferior). Uma heurística poderia ser aplicada no passo intermediário, descartando vizinhos que fogem de um padrão pré-estabelecido ou que não possuam a nota que se busca predizer (a fórmula de correlação necessita que todos os vizinhos utilizados possuam a nota do produto que se busca fazer a predição). De acordo com Adomavicius e Tuzhilin (2005), pode-se, também, calcular a similaridade de todos os usuários com antecedência e recalcular apenas de vez em quando, pois os pares não costumam mudar drasticamente em um curto espaço de tempo.

Também está presente nesta técnica o problema do cold start ou novo usuário. Quando este não possui muitos itens avaliados, o sistema não consegue encontrar usuários similares. Isso se deve ao fato do cálculo da correlação do coeficiente de Pearson necessitar que todos os usuários vizinhos tenham itens em comum para a predição de um item. Algumas estratégias foram propostas para superar esse problema, como a criação de agentes autônomos e a utilização de promoções ou alguma forma de incentivo para aumentar a participação do usuário. Dificuldade similar enfrenta o novo item quando há a sua inclusão no sistema e não é possível recomendá-lo até que algum usuário avalie o mesmo.

Os sistemas baseados em FC também são muito suscetíveis à falhas, segundo Sandvig et al (2007), pois sua busca é baseada em usuários singulares que possuam similaridade, não havendo nenhuma filtragem prévia para verificar se o usuário é um ruído na base de dados. Uma alternativa citada pelo autor é a utilização de regras de associação que, por tentar identificar padrões na base de dados, acabam não sendo tão afetadas por amostras pequenas de ruído. Essas regras, depois de extraídas, poderiam ser aplicadas como heurística no passo intermediário, como comentado previamente.

A esparcialidade de uma base de dados é representada pela quantidade de itens não avaliados, ou seja, as células em branco na matriz Usuário X Item. Quanto mais alta a esparcialidade, maior também é a dificuldade de encontrar usuários similares e de consequentemente gerar recomendações utilizando a FC.

16.4.2.3. Filtragem Baseada em Informação Contextual

Adomavicius e Tuzhilin (2011) afirmam que contexto é um conceito estudado por diferentes disciplinas como Ciência da Computação, Filosofia, Psicologia, etc. Cada área tende a construir sua própria visão, que é normalmente um pouco diferente das outras.

A fim de trazer uma visão mais genérica, Dourish (2004) introduz a taxonomia de contextos, classificando-a em duas visões: representacional e interacional. A primeira visão, representacional, define contexto como um conjunto de atributos observáveis, possíveis de serem descobertos *a priori* e que não variam significantemente sua estrutura no tempo. Em contraste, a visão interacional assume que o comportamento do usuário é induzido por certo contexto, não sendo necessariamente observável.

Focando em Sistemas de Recomendação e nas áreas de pesquisa relacionadas, vários atributos podem ser utilizados para compor um contexto, como por exemplo, a intenção ao realizar uma compra, localização, identidade de pessoas próximas, objetos ao redor, estação do ano, temperatura, status emocional, personalidade, tempo e companhia do usuário. O contexto também pode ser aplicado ao usuário ou à aplicação.

A informação contextual, devido a sua complexidade, pode ser composta por uma estrutura hierárquica possível de ser representada em formato de árvores (Adomavicius e Tuzhilin 2011). Um exemplo de hierarquia de contexto relacionado a programa televisivo: Programa: Programa → Subgênero → Gênero → Canal; Tempo: Hora → Data → Dia da Semana → Mês → Ano.

Em vários domínios, como por exemplo recomendar um pacote de férias, um conteúdo personalizado em um site *Web*, produtos em uma loja *online* ou um filme, seria interessante incorporar informação contextual no processo de recomendação, tais como usuários, itens, tempo e lugar. De acordo com Giordani (2006), a criação de contextos de interesse aumenta as possibilidades de definir o momento mais adequado para recomendar certo produto e, principalmente, quais os produtos que devam compor o perfil do usuário em determinado contexto.

As técnicas de filtragem estudadas aqui utilizam uma função R (Equação 1) e se baseiam exclusivamente nas variáveis usuário e item.

$$R: Usuário \times Item \rightarrow Avaliações \quad (1)$$

A Filtragem Baseada em Informações Contextuais incorpora informação contextual no processo de recomendação, dentro do cenário de decisão do usuário, tais como usuários, itens, tempo e lugar, criando, desta forma, uma função R (Equação 2) com a utilização do contexto:

$$R: Usuário \times Item \times Contexto \rightarrow Avaliações \quad (2)$$

Alguns tipos de informação contextual podem ser mais relevantes para certo tipo de aplicação do que para outros. Para determinar a relevância de certa informação, um especialista no negócio ou um arquiteto de sistemas de recomendação pode manualmente definir a importância, como também utilizar recursos de aprendizado de máquina, mineração de dados e estatísticas sobre as avaliações já existentes. Segundo Giordani (2006), a construção de contextos individuais não é tão simples pois o analista de negócios deveria efetuar uma análise dos dados transacionais de cada cliente em particular, para então tentar identificar possíveis contextos. Neste caso, fica clara a inviabilidade das análises manuais em função da massificação dos dados, tornando necessária a utilização de artifícios que permitam uma descoberta automatizada computacionalmente.

16.4.2. 4. Filtragem Baseada em Outros Contextos

Tkalčič et. al. (2009) afirmam que no contexto de TVD, Sistemas de Recomendação que ignoram as experiências afetivas do usuário enquanto o mesmo está consumindo conteúdo multimídia são no mínimo estranhos, devido ao fato de a indústria do entretenimento estar baseada em causar emoções nas pessoas.

De acordo com a visão de McDonald (2003) a mudança mais importante a se desenvolver na nova geração de Sistemas de Recomendação é a devida complexidade na construção do modelo/perfil de usuário e, o uso apropriado desse modelo. Considerando Perugini et al (2004) modelos/perfis de usuário propiciam indiretamente conexões entre pessoas possibilitando e direcionando a recomendações mais eficientes. Dessa forma, acredita-se que perfis de usuário devem representar diferentes e ricos aspectos da experiência diária de um usuário, considerando a vida real como modelo.

Considere o percurso de um sistema de computador para atingir a mínima compreensão desta otimizada interação de como os humanos procedem nos seus processos de recomendação na "vida real". Note que humanos usam em suas recomendações informações mais complexas que informações efetivamente usadas por um computador. Isto é, usualmente, sistemas computacionais usam informações tais como competências, preferências, informações demográficas dos usuários, entre outras, para a tomada de decisão e posterior recomendação de uma informação, produto ou serviço. Em contraste, humanos, quando tomam decisões, recomendam e personalizam informações, produtos e serviços para outros humanos, além de usar as informações convencionais usadas pelos sistemas computacionais tendem, também, a usar informações adicionais relacionadas a habilidades sociais e psicológicas humanas, tais como, traços de personalidade e Emoção (Nunes and Aranha 2009).

Mesmo sabendo ser impossível perfeitamente antecipar as necessidades humanas individuais para recomendar o produto certo, sabe-se que quanto mais ricas forem as informações sobre o usuário, mais precisos serão os produtos, serviços e /ou pessoas recomendadas. Basta observar o que ocorre na vida real.

Nessa linha, note que Nunes (Nunes 2009) expande o modelo inicial de técnicas de recomendação proposto por Gonzalez et al (2007). Gonzalez por sua vez expande o modelo proposto por Burke (2002). O modelo de Burke (2002) categoriza as técnicas de recomendação em cinco: baseada em conteúdo, filtragem colaborativa, demográfica, baseada em conhecimento e baseada em utilidade. Gonzalez et al (2007) por sua vez cria uma nova categoria intitulada recomendação baseada em outros contextos, incluindo aspectos Psicológicos, tais como, Inteligência Emocional e Interação Social, usando como fonte um Perfil de usuário baseado em contextos diferenciados dos tradicionais modelos demográficos ou baseados em preferência do usuário.

Considerando esse aspecto Nunes (2009) ao expandir o modelo proposto por Gonzalez et al (2007), apresenta a personalidade do usuário como um fator relevante na otimização das recomendações ao usuário, como apresentado na Figura 16.3.

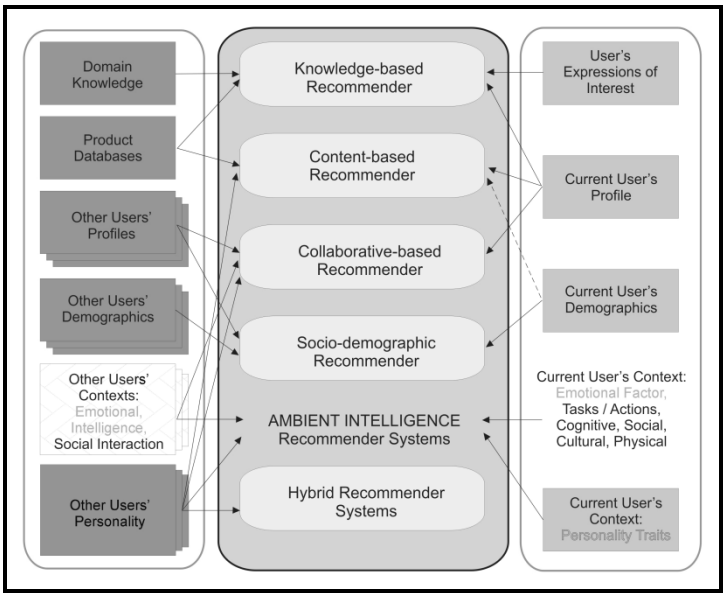


Figura 16.3. Sistemas de Recomendação baseado em personalidade (Nunes 2009)

A técnica de recomendação baseada em personalidade geralmente é aplicada juntamente a outra técnica como a filtragem colaborativa e/ou baseada em conteúdo, representando uma técnica de recomendação híbrida.

Note que para que seja viável a recomendação baseada na personalidade do usuário necessita-se a representação da personalidade, cada projetista de sistema pode usar uma abordagem diferenciada.

Cazella et al (2010) afirmam que a técnica baseada em personalidade geralmente é aplicada juntamente a outra técnica como a filtragem colaborativa e/ou baseada em conteúdo, constituindo-se numa técnica de recomendação híbrida.

16.4.2.5. Filtragem Híbrida

A filtragem híbrida busca unir múltiplas técnicas de recomendação a fim de aproveitar o melhor delas (Sandvig et al 2007). É possível, ainda, combinar duas diferentes técnicas do mesmo tipo, como por exemplo, duas diferentes técnicas baseadas em conteúdo. Uma outra alternativa, é manter os perfis dos usuários baseados em uma análise de conteúdo, comparando-os diretamente, por meio de uma recomendação colaborativa, para determinar possíveis semelhanças (Herlocker 2000; Ansari 2000).

Segundo Herlocker (2000), todas as técnicas baseadas em conhecimento (baseada em conhecimento, baseada em conteúdo, colaborativa e demográfica) sofrem com o problema do *cold start* de uma forma ou outra, seja para tratar de novos itens ou novos usuários. Porém, a junção de técnicas de tipos diferentes é indicada para tentar resolver este problema (Sandvig et al 2007).

Com o uso da filtragem híbrida e utilizando os benefícios da filtragem colaborativa e da filtragem baseada em conteúdo, consegue-se a descoberta de relacionamentos entre usuários, a recomendação de itens baseado na avaliação dos mesmos, o tratamento de

usuários incomuns e a precisão independente do número de usuários e avaliações (Cazella 2006).

16.4.3. Algoritmos de recomendação

A seguir serão descritos brevemente dois algoritmos aplicados a sistemas de recomendação.

16.4.3.1. Algoritmo para Filtragem baseada em conteúdo

A Filtragem Baseada em Conteúdo parte do princípio de que os usuários tendem a interessar-se por itens similares aos que demonstraram interesse no passado. Desta maneira, é definida a similaridade entre os itens. Em alguns casos, pode haver maior dificuldade para estabelecer esta similaridade (Adomavicius et al 2005). Para que seja estabelecida a similaridade entre itens como roupas e brinquedos, por exemplo, seria necessária a identificação dos atributos nos itens a serem comparados (peso, preço, marca, por exemplo). No caso dos itens serem artigos científicos (ou documentos), o processo de comparação pode ser focado nos termos que compõem estes textos (Selton e McGill 1983).

Para que seja estabelecida a similaridade entre artigos (ou documentos), este processo de comparação pode ser facilitado, pois documentos podem ser considerados similares se compartilharem termos em comum. Sendo assim, a filtragem baseada em conteúdo é mais indicada para a recomendação de itens textuais, onde o conteúdo é geralmente descrito com *keywords* (Selton e McGill 1983). Um exemplo é o sistema de recomendação Fab, o qual recomendava páginas *web* para usuários. Este sistema representa uma página *web* com os seus 100 mais importantes termos. Similarmente, o sistema Syskill & *Web*ert representa os documentos com as 128 palavras mais significativas (Adomavicius et al 2005).

Outros exemplos de filtragem baseada em conteúdo são sistemas booleanos de recuperação, onde a consulta constitui-se um conjunto de palavras-chave combinadas com operadores booleanos; sistemas de filtragem probabilística, onde o raciocínio probabilístico é utilizado para determinar a probabilidade que um documento possui de atender as necessidades de informação de um usuário; e linguagem natural (Herlocker 2000).

Conforme mencionado anteriormente sistemas de recomendação baseados em conteúdo podem recomendar itens similares a itens que o usuário gostou no passado. Deste modo, vários itens são comparados com itens que foram avaliados positivamente e os mais similares serão recomendados ao usuário. Conforme apresentado por Adomavicius et. al. (2005) pode-se formalizar esta definição estabelecendo *ContentBasedProfile(c)* como sendo o perfil do usuário *c*. Este perfil é obtido através de uma análise do conteúdo dos itens previamente avaliados pelo usuário utilizando técnicas de recuperação de informação. Por exemplo, *ContentBasedProfile(c)* pode ser definido como um vetor de pesos (wc_1,..., wc_k) onde cada peso wc_i denota a importância do termo k_i para o usuário *c* utilizando-se a medida TF-IDF (*term frequency–inverse document frequency*).

Em sistemas de recomendação baseados em conteúdo, a função utilidade *u(c,s)* é geralmente definida conforme a equação (3):

$u(c,s) = score(ContentBasedProfile(c), Content(s))$	(3)

Tanto o *ContentBasedProfile(c)* do usuário c como o *Content(s)* podem ser representados como vetores (TF-IDF) de pesos e termos $\vec{w}_c$ e $\vec{w}_s$. Além disso, a função utilidade *u(c,s)* normalmente é representada, na literatura de recuperação de informação, por algum tipo de pontuação heurística sobre vetores, como por exemplo, a medida de similaridade do cosseno. O cálculo para a medida de similaridade do cosseno pode ser feito de acordo com a equação (4) (Adomavicius et al, 2005):

$u(c,s) = \cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c . \vec{w}_s}{\|\vec{w}_c\|_2 \times \|\vec{w}_s\|_2} = \frac{\sum_{i=1}^{k} w_{i,c} w_{i,s}}{\sqrt{\sum_{i=1}^{k} w_{i,c}^2} \sqrt{\sum_{i=1}^{k} w_{i,s}^2}}$	(4)

Onde *k* é o número total de palavras no sistema. Desta forma, o cálculo de similaridade é realizado computando o cosseno do ângulo formado pelos dois vetores que representam os documentos (termos e freqüências) (Adomavicius et al 2005). Esta abordagem é utilizada no protótipo desenvolvido para calcular a similaridade entre os artigos. Por exemplo, se o usuário *c*, lê muitos artigos relacionados ao tópico Engenharia de Software, as técnicas de filtragem de conteúdo estão aptas a recomendar outros artigos de Engenharia de Software para o usuário *c*. Estes outros artigos são recomendados porque possuem termos em comum, que neste exemplo, são termos do domínio Engenharia de Software. Neste caso, a função *u(c,s)* será alta, pois os pesos calculados para os termos destes artigos (que compartilham termos em comum referente ao assunto Engenharia de Software) serão altos (Adomavicius et al 2005).

16.4.3.2. Algoritmo para Filtragem Colaborativa

A técnica de filtragem colaborativa (FC) possui um modelo conceitual de operação de fácil entendimento, possibilitando analisar itens a serem recomendados sem preocupar-se com o conteúdo destes itens e sim focando nas avaliações dos itens. A operação de um sistema de recomendação por filtragem colaborativa é similar a recomendação verbal de pessoa para pessoa. Os usuários são supridos de recomendação seguindo três etapas:

1) usuário fornece seu perfil de avaliações;

2) a FC identifica usuários com perfis similares (vizinhos);

3) as avaliações dos vizinhos são combinadas para se gerar as recomendações. Com isso, a percepção do usuário dos passos, relacionados anteriormente, afetará sua percepção sobre todo o sistema.

A seguir as etapas serão detalhadas: na Etapa 1, calcula-se o peso em relação a similaridade do usuário-alvo (aquele que deverá receber uma recomendação) e os demais usuários. Isso pode ser calculado aplicando, por exemplo, o coeficiente do Co-seno ou o coeficiente de Pearson (Cazella, et al 2010); na Etapa 2, seleciona-se, a partir dos resultados do cálculo da similaridade, um subconjunto de usuários com maiores níveis de similaridade, que serão denominados vizinhos. Estes vizinhos serão considerados no cálculo da predição do item a ser recomendado; na Etapa 3, normaliza-se as avaliações

fornecidas pelos usuários ao item em análise para recomendação e calcula-se a predição, ponderando-se as avaliações dos vizinhos com seus respectivos pesos de similaridade.

A Tabela 16.1 apresenta na prática como a filtragem colaborativa funciona. Por exemplo, se quisermos recomendar um produto ao usuário Mauro, procuraremos outros usuários com hábitos de consumo semelhantes. No caso, Paulo e João já compraram produtos que Mauro também comprou (*Prod2*). Em seguida, recomendamos a Mauro produtos que estes dois outros usuários possuem, mas que Mauro ainda não possui, como *Prod1* e *Prod5*. A decisão sobre a recomendação destes produtos baseia-se no histórico de avaliações comuns e o valor de predição calculado.

Tabela 16.1. Recomendação baseada em filtragem colaborativa

Usuário	$Prod_1$	$Prod_2$	$Prod_3$	$Prod_4$	$Prod_5$	$Prod_6$
Paulo		x			X	
João	x	x				
Márcia			x	x	X	
Carlos			x			
Ana	x			x		
Mauro	?	x			?	

16.4.3.2.1. Cálculo do Coeficiente de Similaridade

Para o cálculo da similaridade pode-se adotar o coeficiente de Pearson, uma vez que o mesmo é amplamente utilizado em modelos e mede o grau de relacionamento entre duas variáveis, variando de -1 (ausência total de correlação) a 1 (forte correlação). O cálculo deste coeficiente é realizado conforme a equação 5 (Cazella et al 2010):

$$corr_{ab} = \frac{\sum_i (r_{ai} - \overline{r_a})(r_{bi} - \overline{r_b})}{\sqrt{\sum_i (r_{ai} - \overline{r_a})^2 \sum_i (r_{bi} - \overline{r_b})^2}} \quad (5)$$

Sendo o $corr_{ab}$ a correlação do usuário alvo *a* com um determinado usuário *b*; r_{ai}: é a avaliação que o usuário ativo *a* atribuiu para o item *i*; r_{bi}: é a avaliação que o usuário ativo *b* atribuiu para o item *i*; $\overline{r_a}$ é a média de todas as avaliações do usuário ativo *a*, em comum com o usuário *b*; $\overline{r_b}$ é a média de todas as avaliações do usuário ativo *b*, em comum com o usuário *a*.

4.4.3. 2.2. Predição

A predição é feita independentemente do coeficiente utilizado no cálculo da similaridade, pois ela será gerada através de uma média ponderada das avaliações dos vizinhos que obtiveram um coeficiente de similaridade aceitável, ou seja, com limiar igual ou superior, por exemplo, a 0,3. A equação 6 é utilizada para o cálculo da predição (Cazella et al 2010).

$$p_{ai} = \overline{r_a} + \frac{\sum_{b=1}^{n} (r_{bi} - \overline{r_b}) * corr_{ab})}{\sum_{b=1}^{n} |corr_{ab}|} \quad (6)$$

Sendo $corr_{ab}$ é a correlação do usuário alvo a com um determinado usuário b; p_{ai} a predição de um item i para um usuário alvo a; $\overline{r_a}$ é a média de todas as avaliações do usuário alvo a aos itens que foram pontuados por todos os seus usuários similares; r_{bi} é a avaliação que o usuário ativo b atribuiu para o item i; $\overline{r_b}$ é a média de todas as avaliações do usuário b, em comum com o usuário a.

16.5. Exemplos práticos de uso de personalidade para tomada de decisão computacional

Considerando que os estudos realizados nesse capítulo demonstram uma nova tendência à renovação dos Sistemas de Recomendação, onde os aspectos psico-afetivos humanos devem ser considerados na tomada de decisão computacional, aqui apresenta-se alguns exemplos.

16.5.1. *e-commerce, e-services*

16.5.1.1. Hu and Pu (2009a)

Segundo a visão de Pina e Nunes (2011) o trabalho relata um estudo realizado com usuários para elicitar as preferências entre duas abordagens de sistemas de recomendação: baseado em classificação e baseado em teste de personalidade.

Hu e Pu argumentam que tecnologias de recomendação foram incorporadas a uma série de aplicações, especialmente para sites de *e-commerce* como ferramentas poderosas para ajudar a lidar com a sobrecarga de informações de usuários e prestação de serviços personalizados. Diante desta constatação Hu e Pu realizaram um experimento para comparar dois sites de recomendação de filmes. Para tanto foram adotados os sistemas : *MovieLens*[20] e *WhattoRent*[21].

Para execução do experimento foram feitas algumas adaptações na interface do *MovieLens* para que as diferenças existentes entre ele e o *whattorent* não afetasse a satisfação dos usuários no sistemas. Foi adicionado um link atrelado a cada filme recomendado que exibe um pequeno o clipe do filme. O objetivo é de ajudar o usuário no processo de avaliação dos filmes recomendados. Foi também inserido um grupo de botões ao lado de cada filme recomendado para facilitar aos usuários o fornecimento das avaliações em cinco níveis que vão de muito desinteressada até muito interessado.

---

[20] O *MovieLens* é um sistema de recomendação baseado em classificação, foi construído pelo grupo de pesquisa GroupLens liderado por John Riedl da Universidade de Minnesota. O *MovieLens* aprende as preferências dos usuários a partir de suas avaliações e utiliza a tecnologia de filtragem colaborativa para recomendar filmes. Quando um novo usuário faz logon, ele é obrigado a informar pelo menos 15 filmes que já tinha visto. Estas classificações são usadas para construir o perfil de usuário e encontrar os 'vizinhos' que têm gostos similares ao do presente usuário. Com base na classificação a partir desses vizinhos, ele prediz quanto o usuário possivelmente irá gostar de um filme. Em seguida, ele apresenta todas as recomendações de Filmes em ordem decrescente dos escores previsto. Usuários podem refinar os seus perfis de preferência classificando mais filmes e também podem alterar as classificações anteriores em interações futuras com o sistema.

[21] O *whattorent é um* sistema de recomendação baseado em personalidade, foi construído com base na Teoria *LaBarrie* que afirma que os "telespectadores interagem emocionalmente com um filme da mesma maneira que eles interagem com outros seres humanos". Seus desenvolvedores elaboraram um questionário com 20 questões de personalidade onde as preferências dos usuários são decifradas e é feita a recomendação do filme.

O experimento foi feito com 30 participantes, sendo 11 mulheres e 19 homens, distribuídos em quatro grupos etários e com diferentes profissões (cada usuário recebeu umauxilio financeiro como motivação). Cada um dos participantes avaliou os dois sistemas de recomendação (*MovieLen*s e *WhattoRent).*

A execução do experimento foi no gabinete de um administrador que supervisionava a experiência e ajudava os participantes a concluir todas as tarefas com sucesso. A seguir são listados os passos do experimento: (i)Participante registra uma nova conta para evitar quaisquer influências a partir do histórico de uso; (ii) No *MovieLen*s o participante classifica 15 filmes que tenha visto antes e no *whattorent* o participante responde a 20 questões de personalidade; (iii) São respondidas duas perguntas sobre o atual humor do usuário; (iv) O participante classifica 6 filmes recomendados em uma escalas que vai de 1 = muito desinteressado a 5 = muito interessado; (v) O participante responde um questionário de avaliação on-line onde cada pergunta é respondida em uma escala *Likert* sobre as preferências entre os dois sistemas e suas razões.

Os critérios de avaliação utilizados foram: (i) Percepção de Precisão - Mede quanto os usuários sentem que as recomendações correspondem às suas preferências; (ii) Esforço do Usuário - Mede a quantidade de esforço subjetivo que os usuários sentiram que gastaram e o tempo real que foi utilizado para completar processo de preferência; (iii) Fidelidade do Usuário- Avalia se o sistema tem a capacidade de convencer seus usuários a adquirir o produto recomendado, se o sistema gera interesse de reutilização ou o sistema é satisfatório ao ponto do participante fazer uma indicação para os seus amigos; (iv) Os resultados mostram que a percepção da precisão entre os dois sistemas não é significativamente diferente. No entanto, os usuários gastam muito menos esforço para completar o perfil de preferência baseado em personalidade do que no sistema baseado em classificação. Além disso, os usuários expressaram forte intenção de reutilizar o sistema baseado personalidade e apresentá-lo aos seus amigos. Após analisar os resultados do questionário final sobre preferência, foi possível constatar que 53,33% dos usuários (16 em 30) preferiram o sistema baseado em teste de personalidade, enquanto que 13,33% dos usuários (4 em 30) preferiram o sistema baseado em classificação.

As autoras acreditam que a principal contribuição do artigo foi perceber quanto a extração de preferências baseada em abordagens de personalidade pode ajudar aos usuários a revelar as preferências ocultas e assim aumentar a precisão recomendação. Ainda o artigo demonstrou a preferência dos usuários por sistemas de recomendação baseado em personalidade. Uma das razões que justificaram essa preferência foi a redução do esforço. Porém os usuários que preferiram a recomendação convencional justificaram a sua preferência considerando a vantagem importante de ter controle sobre a criação do seu próprio perfil.

16.5.1.2. Hu and Pu (2009b)

Segundo Pina e Nunes (2011) esse trabalho complementa o trabalho apresentado na seção anterior. O artigo avalia a aceitação de usuários aos sistemas de recomendação baseado em personalidade (RBP), usando o modelo de aceitação de tecnologia (TAM).

O artigo utiliza o mesmo experimento da seção anterior porém utiliza uma formalização melhor para referenciar os resultados. O experimento, como descrito anteriormente foi realizado com 30 participantes, sendo que cada um dos participantes avaliou os dois sistemas de recomendação (*MovieLen*s e *WhattoRent).*

A avaliação usou os critérios baseados no modelo TAM, que foi projetado para compreender a relação causal entre variáveis externas de aceitação dos usuários e o uso real do computador, buscando entender o comportamento deste usuário através do conhecimento da utilidade e da facilidade de utilização percebida por ele (DAVIS et *al* 1989). Segundo Davis et *al* (1989) as pessoas tendem a usar ou não uma tecnologia com o objetivo de melhorar seu desempenho no trabalho, esse fator é chamado de utilidade percebida. Porém, mesmo que essa pessoa entenda que uma determinada tecnologia é útil, sua utilização poderá ser prejudicada se o uso for muito complicado, de modo que o esforço não compense o uso, esse fator é chamado de facilidade percebida. É também avaliado o critério de intenções comportamentais para uso. Onde são avaliados três aspectos: intenção de compra, a intenção de voltar ao sistema e a intenção de recomendar este sistema para amigos.

Em relação a utilidade percebida - os filmes recomendados pelo sistema baseados em personalidade foram avaliados como de interesse 54,9%, comparados àqueles recomendados pelo sistema baseado em classificação com 42,7% de interesse. Em relação a facilidade de uso - o esforço global cognitivo percebido é significativamente menor no sistema baseado em personalidade do que no sistema baseado em classificação. Em relação as intenções comportamentais de uso – nos três aspectos os RBP obteve um vantagem. Os resultado final encontrado da comparação de aceitação entre os dois sistemas de recomendação mostram que 53,33% dos usuários preferiram o sistema baseado em personalidade em relação a 13,33% do usuários que preferiram o sistema baseado em classificação.

Assim o artigo permitiu concluir a preferência dos usuários entre sistemas de recomendação baseado em personalidade e sistema de recomendação baseado em classificação. E também revelou percepções sobre o processo de escolha do usuário, mostrando que um design de interface simples, pouco esforço inicial, e recomendação de boa qualidade são os fatores mais influentes que contribuem para a aprovação dos usuários nos sistemas de recomendação.

A pesquisa mostrou que a personalidade é um fator de resistência primária e que determina os comportamentos humanos e que são significativas as ligações entre a personalidade e os interesses das pessoas. Sistemas de recomendação baseados em personalidade podem fornecer informações e serviços mais personalizados, desde que eles compreendam melhor os clientes sob o ponto de vista psicológico.

16.5.1.3 Recio-Garcia et al (2009)

Segundo Pina e Nunes (2011) o artigo apresenta um novo método de realizar recomendações para grupos usando a técnica de filtragem colaborativa e considerando a

composição da personalidade do grupo. A abordagem utiliza personalidades de membros do grupo para escolher o filme mais interessante que pode melhor satisfazer todo o grupo. A partir dos algoritmos de recomendação clássicos foram feitas três adaptações diferentes utilizando como base o Peso Modo Conflito (CMW) aplicado aos membros do grupo. A pesquisa foi testada no domínio de recomendação de filmes, usando os dados *MovieLens* em conjunto com grupos de usuários de diferentes tamanhos e graus de homogeneidade.

Os autores relatam que conflito é uma parte natural de nossas interações com os outros. Diferentes pessoas têm diferentes expectativas e desejos que geralmente parecem ser incompatíveis. Em situações de conflito, pode-se descrever o comportamento de um indivíduo sob duas dimensões básicas: (i) Assertividade - medida em que a pessoa tenta satisfazer suas próprias preocupações; (ii) Cooperativismo - medida em que a pessoa tenta satisfazer as preocupações das outras pessoas.

O teste TKI (Instrumento Modo Conflito Thomas-Kilmann) foi escolhido pois se concentra na gestão de conflito. O TKI constrói um perfil do usuário por meio de 30 perguntas de escolha única. O teste fornece pontuações para enquadrar o usuário dentre os modos: Competitivo, Colaborativo, Acomodado, Comprometido e Cauteloso. Os cinco modos representam as preferências do indivíduo quando têm que enfrentar situações de conflitos. Estas pontuações são normalizadas utilizando uma amostra de 8.000 pessoas. O Valor do Peso Modo Conflito (CMWU) representa o comportamento do indivíduo após a avaliação TKI. É calculado usando a seguinte equação: CMW = 1 + Assertivo - Cooperativismo. A seguir a Tabela 16.2 demonstra a escala de pontuação dos cinco modos utilizada para extrair o nível assertivo ou cooperativo do indivíduo para serem utilizados na equação para encontrar o CMWU:

Tabela 16.2. Tabela dos coeficientes para calcular CMW Fonte: (Recio-Garcia *et a*l 2009)

	Assertiveness		Cooperativeness	
TKI Mode	High	Low	High	Low
Competing	0.375	-0.075	-0.15	0
Collaborating	0.375	-0.075	0.375	-0.075
Compromising	0	0	0	0
Avoiding	-0.375	0.075	-0.375	0.075
Accommodating	-0.15	0	0.375	-0.075

No contexto de recomendações para grupos as principais abordagens para gerar preferência de grupos com base nas preferências de um usuário são: (i)Fusão das recomendações feitas para os indivíduos (união ou interseção); (ii) Agregação de indivíduos para classificação; (iii) Construção de um modelo de preferência do grupo. Nesse estudo foi escolhida a abordagem (ii) para recomendação de grupo onde se assume que um indivíduo fornece previsões de classificação para cada usuário do grupo e para cada item a ser recomendado.

Foram implementados três algoritmos de recomendação de grupo a partir do algoritmo básico para que incluíssem o CMW de uma forma diferente. Posteriormente foram feitas comparações entre s resultados das três versões. Os algoritmos são: (i) Minimizando penalização - Este algoritmo utiliza o CMW para calcular o quão interessante

é o melhor conjunto de itens candidatos da recomendação de um membro para as outras pessoas do grupo; (ii) Média de satisfação – Este algoritmo é inspirado no procedimento médio de satisfação, o critério de seleção escolhe n itens com maior média de satisfação; (iii) Menor sofrimento – Este algoritmo propõe representar uma variação do procedimento de minimização do sofrimento empregados pelas classificações inferida pelo *MovieLens* para gerar a agregação de preferências.

Os autores apresentaram um experimento com um conjunto de 70 alunos no domínio de recomendação de filmes. Os alunos preencheram suas preferências sobre os filmes dentre uma lista com 50 filmes heterogêneos selecionados a partir do conjunto de dados do *Movielens*. Em média os usuários avaliaram 33 filmes. Em um outro momento os alunos fizeram o teste TKI, possibilitando o calculo do Valor Peso Modo Conflito (CMW) para cada indivíduo. Na seqüência o CMW é aplicado nos três algoritmos já citados para obter as recomendações para os grupos.

Para avaliar os resultados foi feita uma comparação usando uma medida simples de avaliação que conta quantos filmes classificados pelo usuário estão no conjunto dos melhores filmes sugeridos pelo sistemas de recomendação. Ao analisar os resultados dos algoritmos com a aplicação do fator CMW foi possível constatar que o algoritmo minimização penalizações funciona melhor propondo apenas um filme. Com a adoção do algoritmo média de satisfação e menor sofrimento é refletida uma melhoria geral na precisão da recomendação. Os resultados também melhoram quando aplicados em grupos com pessoas que tenham soluções de conflito de personalidades heterogêneos.

O artigo contribui demonstrando que sistemas de recomendação para grupos poderiam ser melhorados em até 7% de precisão ao utilizar os valores da personalidade dos indivíduos obtidos a partir do TKI e CMW.

16.5.2. TV digital (Trevisan 2011)

Uma aplicação para recomendação de programas de TV (Trevisan 2011) pode rodar em diferentes aparelhos eletrônicos: celulares, computadores e televisores, entre outros com algum tipo de acesso a rede mundial de computadores. Porém, o mais usual é disponibilizar este tipo de aplicação em um aparelho de televisão. Com o surgimento da TV Digital no Brasil e a crescente expansão do sinal, a possibilidade do desenvolvimento de programas interativos tornou acessível a criação deste tipo de sistema. Então, utilizando o SBTV, foram testadas as duas plataformas do Ginga (Ginga NCL e Ginga-J) para o desenvolvimento do protótipo da aplicação. O Ginga NCL foi o que apresentou maior maturidade e que possui o maior número de aplicações sendo desenvolvidas no cenário nacional, até por conta da maior maturidade que possui. Os testes foram realizados no emulador XletView que utiliza uma implementação de referência do MHP (*middleware* do padrão europeu). O XletView é um emulador de TVD baseado no *middleware* MHP que permite a execução de xlets em um ambiente PC. É um projeto de *software* livre e está sob a licença GNU GPL. Um xlet é uma aplicação Java para TVD e equivale a um applet Java em um PC (Araujo e Carvalho 2009).

O protótipo em TVD implementado acessa um servidor HTTP e busca as recomendações disponíveis para o usuário que o está utilizando. O telespectador pode identificar o quanto o programa recomendado atende às suas necessidades ou desejos. Esta informação alimenta o sistema como uma nova avaliação, aumentando a precisão de futuras sugestões. A base de dados de programas de televisão foi disponibilizada pela empresa Revista Eletrônica para o presente trabalho e pôde ser obtida via FTP, no formato XML. A grade de programação utilizada no protótipo foi extraída entre 21/06/2010 a 05/07/2010 (quinze dias). A fim de viabilizar o experimento e devido à quantidade de usuários e o tempo de experimentação, foi decidido pela importação de somente programas dos canais abertos da televisão brasileira: o SBT, a RBS, a Bandeirantes, a TV COM, a TV Futura e a Rede TV. Desta forma, foi obtida uma base de dados com uma esparcialidade não tão alta e com programas mais conhecidos.

O contexto cujo usuário está inserido no momento da recomendação é informado pelo mesmo de forma explícita a partir de uma lista pré-definida, porém configurável. Estes contextos foram elencados empiricamente a partir da análise dos gêneros, subgêneros e classificação etária dos programas disponíveis na grade de programação. Como os contextos são configuráveis, é possível a aplicação de um algoritmo de mineração de dados no momento em que há a posse de uma base de dados com uma quantidade maior de avaliações e extrair novos contextos, talvez mais significativos. Os contextos elencados para a execução do protótipo foram por exemplo, 1) Nome do contexto: Sozinho; Descrição: Adulto assistindo sozinho; Aparelho: Televisor; 2) Nome do contexto: Com Crianças; Descrição: Acompanhado de crianças; Aparelho: Televisor; 3) Nome do contexto: Recebendo Amigos; Descrição: Recebendo amigos para um jantar; Aparelho: Televisor; 4) Nome do contexto: Com Namorado(a); Descrição: Acompanhado do namorado(a)/esposo(a); Aparelho: Televisor; 5) Nome do contexto: Locomovendo-se; Descrição: Viajando de ônibus para o trabalho; Aparelho: Celular.

A biblioteca Apache Mahout foi utilizada como *framework* para a construção das recomendações. Além de vários outros recursos, ela fornece algoritmos de filtragem colaborativa e baseada em conteúdo, estando sob a licença de software da Apache (Mahout, 2011). Os testes de personalidade foram realizados utilizando a aplicação *Personality Inventory PV1.0* Nunes et al (2010). Esta é uma aplicação *online* que utiliza os testes NEO-IPIP e TIPI. Cada questão respondida tem um valor atribuído entre 1 e 5 (Figura 16.1c). A partir destes valores, é realizado um cálculo que gera um valor entre 1 e 100 para cada um dos itens do Big Five.

A seleção do contexto em que o usuário se encontra é realizada de forma explícita pelo mesmo antes de receber uma recomendação. Pode ser alterada a qualquer momento, pressionando a tecla verde do controle remoto. Quando a lista é exibida, basta clicar com a tecla de comando “para cima” e “para baixo” do controle remoto até posicionar sobre o contexto desejado e, então, pressionar o botão “ok” do mesmo, conforme a Figura 16.4.

Uma vez definido o contexto do usuário, o sistema acessa o servidor e busca uma lista de recomendação para o usuário. É possível que nenhum programa seja recomendado

para o usuário, neste momento, pelo algoritmo descrito no modelo do sistema anteriormente. Para que o usuário não fique sem receber nenhuma recomendação, o sistema busca por programas que se enquadrem no contexto selecionado e que possua os gêneros, subgêneros e classificações etárias melhor avaliadas por este usuário no passado.

Neste ponto, o sistema acaba realizando uma filtragem baseada em conteúdo e, desta forma, o usuário não fica frustrado por não receber nenhum item recomendado e acaba com o problema do *cold start*. Outro ponto positivo de recomendar itens mesmo que o algoritmo de filtragem colaborativa não retorne nenhuma recomendação é a alimentação do sistema com o *feedback* do usuário que é enviado para cada recomendação gerada, o que de certa forma contribui para o sucesso das suas próximas utilizações.

A lista de itens recomendados apresenta o título do programa, o ícone do canal, a classificação etária e a avaliação dada pelo usuário representada pelas estrelas ao lado de cada programa (Figura 16.5). Para navegar pelos itens e avaliá-los, basta que o usuário utilize as teclas de navegação do controle remoto "para cima" e "para baixo" a fim de posicionar-se no item desejado, e a tecla "para esquerda" e "para direita" para aumentar ou diminuir a nota dada para o programa recomendado.

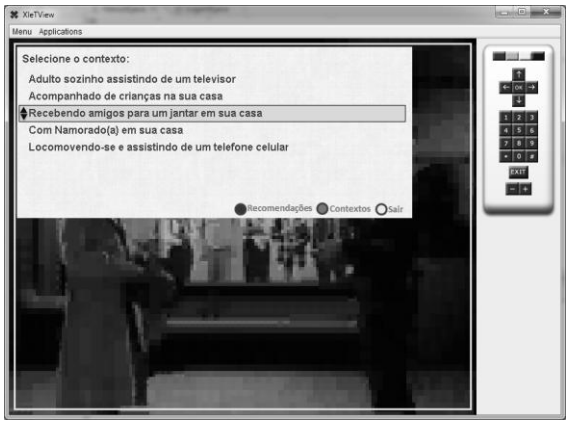


Figura 16.4 – Seleção explícita de contexto(Trevisan 2011).

Figura 16.5 – Lista de recomendações fornecidas pelo TvPlus (Trevisan 2011).

Cada avaliação realizada pelo usuário pela TV é enviada para o servidor e armazenada na tabela de avaliações. Quando o algoritmo de recomendação for executado novamente por qualquer usuário do sistema, estas informações transmitidas no *feedback* realizado serão processadas e poderão gerar uma recomendação diferente das anteriormente criadas.

16.5.3. Outros

16.5.3.1. Presidencial Francesa (Nunes 2009)

Nunes (2009) apresenta um protótipo de Sistema de Recomendação que objetiva dar indícios do quão importante os aspectos psicológicos, como traços de personalidade, são no processo de tomada de decisão computacional, isto é, no processo de recomendação (em Sistemas de Recomendação e/ou Sistemas de Combinação Social).

A Presidencial Francesa é um Sistema de Recomendação mostrando sua habilidade para recomendar pessoas, nesse caso, consideradas no contexto de produtos e não prestadoras de serviço[22].

Cenário: O cenário desse sistema é apresentado pelas "eleições Presidenciais na França" realizada em 2007. Nesse caso, o Sistema de Recomendação foi usado para gerar uma recomendação considerando a melhor escolha de um candidato para uma pessoa votar, considerando a reputação dos candidato sob as vistas da própria pessoa que receberia a recomendação. Esse experimento foi aplicado de dezembro de 2006 a julho de 2007.

Método: Objetivando-se criar um Perfil de usuário adequado à recomendação, esse experimento usou a abordagem de traços de personalidade (já apresentado na seção anterior) proposto por Nunes (Nunes 2009). Então, o Perfil/Reputação Psicológico do Usuário foi criado considerando o teste de personalidade NEO-IPIP (Johnson 2001) baseado em 300 questões.

Cerca de 100 pessoas foram convidadas a participar do experimento. Cada pessoa que participou do experimento foi instruída a responder o NEO-IPIP 3 vezes: (i) uma para o "candidato ideal", as respostas refletiam como cada pessoa via o que considera características psicológicas de um presidente ideal; (ii) outra para o candidato "Nicolas Sarkozy", isto é, cada pessoa respondeu o questionário informando como via e o que pensava sobre as características de tal candidato (Reputação); (iii) e outra para a candidata "Ségoléne Royal", isto é, cada pessoa respondeu o questionário informando como via e o que pensava sobre as características de tal candidato (Reputação).

Através das respostas, a modelagem dos aspectos psicológicos dos candidatos, isto é, o Perfil de usuário (Reputação), foi criado. A recomendação[23] foi feita baseada nesses aspectos psicológicos.

Objetivando validar a precisão da recomendação gerada pelo Sistema de Recomendação, foi requisitado a cada pessoa que participou do experimento gentilmente confirmasse se o resultado da recomendação realmente havia representado o seu *voto real.*

Resultados: Apenas 10% das pessoas requisitadas responderam totalmente o questionário. O Sistema de Recomendação forneceu 2 tipos de recomendação. A primeira recomendação foi baseada nas 30 facetas do Big Five (para mais informação ver Nunes 2009) e, a segunda foi baseada simplesmente nas 5 dimensões do Big Five.

---

[22] O Sistema de Recomendação gera uma pessoa com um "produto", pois, nesse caso, a pessoa é considerada como um pacote fechado, ou seja, a pessoa significa verdadeiramente o nome dessa pessoa. Nesse caso, a pessoa não é considerada como um provedor de serviço, como normalmente as pessoas são consideradas. A visão de produto significa que a pessoa recebe uma recomendação de alguém constando um nome a ser considerado como suporte a sua tomada de decisão, diferentemente da visão de serviço. Na visão de serviço, a pessoa recebe um nome para ser usado como um provedor de serviço, isto é, este "nome"(pessoa) irá provavelmente executar algum serviço a posteriori. Na visão de produto, o Sistema de Recomendação gera uma resposta passiva, diferente da visão de serviço onde a resposta, provavelmente, gera uma interação dinâmica com a pessoa que executará o serviço.

[23] O Sistema de Recomendação usou a técnica de recomendação baseada em outros contextos, considerando a personalidade. A abordagem usada na implementação foi algoritmo de busca linear usando a técnica do vizinho mais próximo.

Os resultados da recomendação foram muito mais satisfatórios e representativos que o esperado. Os resultados são: (i) considerando a primeira recomendação, mais precisa (baseada nas facetas), a recomendação foi 100% precisa. Isto é, todas as recomendações geradas pelo Sistema de Recomendações combinaram com os votos efetivos dos participantes; (ii) considerando a segunda recomendação, menos precisa (baseado nas dimensões), a recomendação atingiu 80% de acerto. Isto é, houve 20% de casos onde a recomendação gerada pelo Sistema de Recomendação foi incompatível com o voto efetivo dos participantes.

O resultado demonstrou que usando uma complexidade maior na representação do Perfil Psicológico do usuário, no caso das facetas, o resultado obtido foi muito satisfatório.

O sucesso desse experimento referenciou o grande potencial no uso de questões psicológicas para auxiliar na tomada de decisão computacional através do uso de Sistemas de Recomendação. Ele foi o indicio necessário para apostar nessa nova e potencial tendência a ser usadas nas futuras recomendações.

16.6. Conclusões

Considerando que os Sistemas de Recomendação constituem-se em uma área de pesquisa que apresenta uma série de desafios e oportunidades. Já faz algum tempo que as pesquisas na área saíram da academia e tomaram forma no mercado, sendo inserido como solução em grandes sites de *e-commerce* como *Amazon.com,* Submarino.com, e atualmente o próprio site de recuperação de informação Google, implementou uma solução de recomendação, onde o usuário uma vez logado poderá recomendar uma página.

Vimos neste capítulo como os Sistemas de Recomendação dotados de traços de personalidade podem ser utilizados para que se possa conhecer melhor os hábitos de consumo e interesses dos usuários, e como este tipo de conhecimento pode ser empregado para fidelizar clientes *web*.

Ao longo deste capítulo foi introduzida a área Computação Afetiva, incluindo uma breve descrição dos aspectos que envolvem a afetividade enfatizando a Personalidade. Seguindo-se por uma descrição e exemplificação de como, porque e quando a Computação Afetiva, principalmente a personalidade, potencializa a tomada de decisão humana. Apresentou-se, também, as abordagens de personalidade existentes exemplificando as codificáveis em computadores. Discutiu-se como os aspectos de personalidade influenciam na identidade do usuário e como isso afeta seu perfil. Na seqüência, foi descrito os critérios de armazenamento bem como as metodologias existentes hoje para extração de personalidade por computadores. Ainda discutiu-se as formas existentes de tomada de decisão computacional enfatizando os Sistemas de Recomendação, focando em sua aplicação, principalmente em e-commerce e TV digital. Questões relativas a técnicas e estratégias de recomendação foram apresentadas com o resumo de seus algoritmos.

Esse estudo tem mostrado que empresas tem buscando cada vez mais personalizar a relação humano-computador e esta tendência é motivada principalmente pelos interesses do *e-commerce e e-services.* Porém para evoluirmos na área da Computação Afetiva é preciso encontrar as melhores técnicas para capturar a personalidade e o perfil dos

usuários. Existem linhas de pesquisa na área da psicologia que acreditam que é possível definir um indivíduo pela sua personalidade, a partir da análise de suas características.

Com o propósito de definir a personalidade surgem várias abordagens, tais como humanística, cognitiva, tipos, traço entre outras. Com base nessas abordagens são construídos modelos para capturar a personalidade. Os trabalhos estudados em sua maioria fizeram uso dos modelos baseados no BIG FIVE/FFM o que aparenta ser uma tendência, visto que, trabalha com a abordagem de traços. As pesquisas, que de modo geral utilizaram experimentos com um número pequeno (em média 60, entre homens e mulheres) de pessoas, demonstram que estes inventários são extremamente necessários à evolução da classificação das recomendações, porém, há estudos que comprovam que estas técnicas são formas invasivas e que desestimulam usuários a respondê-los, por seu extenso volume de questionamentos. Com isso, novos modelos têm sido sugeridos com a finalidade de melhorar a usabilidade do preenchimento destes formulários, associados às ferramentas probabilísticas capazes de otimizar a recomendação, ressaltando a necessidade de pesquisa com outras técnicas estatísticas que se adéquam ao contexto.

Dispositivos de aquisição de dados como teclado e mouse são soluções que não oneram os sistemas e nas pesquisas apresentadas, tem obtido resultados importantes na definição da personalidade e dos estados emocionais. Sensores de detecção de sinais fisiológicos, apesar de onerosos, possuem alta precisão na detecção destas atividades fisiológicas e melhoram bastante a classificação dos estados emocionais. Vimos, também a possibilidade de reconhecer e extrair traços de personalidade através de outra abordagem como diálogos e o uso de modelos estatísticos advindos da técnica de reconhecimento de padrões. Notoriamente, a importância de se reconhecer e responder traços de personalidade automaticamente possibilitaram a automatização nos processos de comunicação, derivando aplicabilidade variada em áreas da ciência, tecnologia e social.

O Sistema de Recomendação é uma tendência mundial quando estamos tratando de problemas que necessitam de apoio à decisão. Os Sistemas de Recomendação, de uma forma geral, tentam melhorar suas recomendações utilizando formulários de intervenção a usuários que buscam serviços e produtos.

Contudo, evidencia-se que um Sistema de Recomendação associado à Filtragem Colaborativa, onde as técnicas probabilísticas garantem uma maior acurácia à recomendação pela medida de similaridade dos usuários, tendo ainda Hardware e Software agregados aos sistemas com finalidades de enriquecer as definições dos modelos de usuário, modelos de itens e modelos de personalidades, são desejos da comunidade de Computação Afetiva na tentativa aproximar cada vez mais a um cenário real.

Alguns desafios emergem neste contexto, tais como: uso da computação afetiva com a utilização de agentes de software ou agentes de comunicação para uma maior interação com o usuário; criação de sistemas eficazes para recomendação a partir da aplicação de traços de personalidade adaptando a interface às reais necessidades psico-cognitivas do usuário, entre desafios que estão ainda surgindo dentro de uma área extremamente promissora no que tange a pesquisa.

16.7. Agradecimentos

Gostaríamos de agradecer a todos os alunos (IC, IT, TCC e Mestrado) que orientamos na área de Sistemas de Recomendação e contribuíram na elaboração do material de pesquisa usado na composição desse capítulo, principalmente ao aluno Luiz Trevisan que trabalhou o protótipo TV Plus, o aluno Jonas Santos Bezerra que está desenvolvendo o PersonalityML e a aluna Sandy Porto + Wanderson Costa que estão desenvolvendo um aplicativo para a extração automática de personalidade.

16.8. Referências Bibliográficas

Adomavicius, Gediminas E Tuzhilin, Alexander. (2011) Context-Aware Recommender Systems. In: Ricci, Francesco Et. Al. Recommender Systems Handbook. Cap. 7, P. 217-253.

Adomavicius, Gediminas E Tuzhilin, Alexander. (2005) Toward The Next Generation Of Recommender Systems: A Survey Of The State-Of-The-Art And Possible Extensions. In: Ieee Transactions On Knowledge And Data Engineering, P. 734-749, Los Alamitos.

Allport, G. W.(1927). Concepts of trait and personality. Psychological Bulletin, (24):284–293. (Available at http://psychclassics.yorku.ca/Allport/concepts.htm).

Andrade, L.O. S.; Nunes, M. A. S. N. (2011) Computação Afetiva: uma breve análise sobre os Sistemas de Recomendação Baseado em Conteúdo e Sistemas de Recomendação de Filtragem Colaborativa. Relatorio Técnico- Notas de Mestrado. PROCC-Universidade Federal de Sergipe.

Ansari, Asim; Essegarier, Skander E Kohli, Rajeev. (2000) Internet Recommendation Systems. In: Journal Of Marketing Research, P. 363-375, Chicago.

Araujo, Sandra R. C. E Carvalho, Victor T. (2011) Emuladores Para Tv Digital - Openmhp E Xletview. Disponível Em: < http://Www.Tvdi.Inf.Br/Upload/Artigos/Artigo7.pdf> Acesso Em: 10 Mar.

Balabanovic, Marko E Shoham, Yoav Fab. (1997) Content-Based, Collaborative Recommendation. In: Communications Of The Acm, P. 66-72, Nova Iorque.

Bezerra, J. S., Nunes. M. A. S. N., Oliveira. A. A. (2011) Desenvolvimento de metodologias de extração de perfil psicológico de usuário para aplicação em Sistemas de Recomendação objetivando personalização de produtos e serviços em e-commerce. Relatório Técnico de Pesquisa, Universidade Federal de Sergipe.

Boyd, D. (2002). Faceted id/entity: Managing representation in a digital world. Master's thesis, Cambridge, MA.

Burger, J.M. (2000). Personality. Wadsworth, fifth edition.

Burke, Robin. (2002) Hybrid Recommender Systems: Survey And Experiments. In: User Modeling And User-Adapted Interaction, P. 331-370, Hingham.

Carreira, R., Crato, J.M., Gonçalves, D. and Jorge, J. A. (2004). Evaluating adaptive user profiles for news classification. In IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces, pages 206–212, New York, NY, USA. ACM Press.

Cazella S. C., Nunes, M. A. S. N., Reategui, E. A . A. (2010) Ciência Da Opinião: Estado Da Arte Em Sistemas De Recomendação. In: Wagner Meira Jr. E André C. P. L. F. De Carvalho(Org.).. (Org.). Jai: Jornada De Atualização Em Informática Da Sbc. Rio De Janeiro: Editora Da Puc Rio, V. , P. 161-216.

Cazella, S. C.; Correa, I. ; Barbosa, J. ; Reategui, E. (2009) Um Modelo Para Recomendação De Artigos Acadêmicos Baseado Em Filtragem Colaborativa Aplicado À Ambientes Móveis. Revista Novas Tecnologias Na Educação, V. 7, P. 12-22.

Cazella, S. C. (2006) Aplicando A Relevância Da Opinião De Usuários Em Sistema De Recomendação Para Pesquisadores. Tese (Doutorado Em Ciência Da Computação) - Universidade Federal Do Rio Grande Do Sul, Porto Alegre, RS.

Chanel, G. (2009) Emotion assessment for affective computing based on brain and peripheral signals. Thèse de doctorat : Univ. Genève.

Damasio, Antonio R. (1994) Descartes' Error: Emotion, Reason, And The Human Brain. Quill, New York.

Davis, F. D.; Bagozzi, R. P. and Warshaw, P. R.. (1989). User acceptance of computer technology: a comparison of two theoretical models. Manage. Sci. 35, 8 982-1003.

Donath, J. S. (2000). Being Real: Questions of Tele-Identity. In: Ken Goldberg, editor, The Robot in the Garden: Telerobotics and Telepistemology in the Age of the Internet, chapter 16, pages 296–311. The MIT Press, first edition.

Donath, J.S. (1999) Identity and Deception in the Virtual Community. In M. A. Smith and P. Kollock, editors, Communities in Cyberspace, chapter 2, pages 29–59. Routledge, London, first edition.

Dourish, Paul. (2004) What We Talk About When We Talk About Context. In: Personal And Ubiquitous Computing. P. 19-30.

Dunn, G., Wiersema, J., Ham, J., and Aroyo, L. (2009) Evaluating Interface Variants on Personality Acquisition for Recommender Systems. In *Proceedings of the 17th international Conference on User Modeling, Adaptation, and Personalization: Formerly UM and AH* .G. Houben, G. Mccalla, F. Pianesi, and M. Zancanaro, Eds. Lecture Notes In Computer Science, vol. 5535. Springer-Verlag, Berlin, Heidelberg, 259-270. (2009)

Erikson, Erik H.(1980). Identity and the Life Cycle. Norton.

Khan, I., Et Al. (2008) Measuring Personality from Keyboard and Mouse Use. ACM International Conference Proceeding Series, Vol. 369. Portugal.

Giddens, A. (1991) Modernity and Self-Identity. Self and Society in the Late Modern Age. Stanford university Press, Stanford, California.

Gill, A. J. ; Oberlander, J. (2003). Perception of e-mail personality at zero acquaintance: Extraversion takes care of itself; Neuroticism is a worry. Proceedings of the 25th Annual Conference of the Cognitive Science Society (pp. 456–461). Hillsdale, NJ: LEA.

Giordani, Alexandre. (2006) Wine Bunch: Uma Ferramenta Baseada Em Agrupamento Para Auxiliar Na Identificação De Contextos De Interesses De Consumidores Em *E-commerce*. Trabalho De Conclusão De Curso (Graduação Em Análise De Sistemas) - Universidade Do Vale Do Rio Dos Sinos, São Leopoldo, RS.

Goffman, E. (1959). The Presentation of Self in Everyday Life. Anchor Book.

Goldberg, L. R., Johnson, J. A., Eber, H. W., Hogan, R., Ashton, M. C., Cloninger, R. C., Gough, H. G., (2006) The international personality item pool and the future of public-domain personality measures. Journal of Research in Personality 40 (1), 84-96.

Gosling, S. (2008) Psiu, Dê Uma Espiadinha! *Editora Campus*.

Gonzalez, G., De La Rosa, J.L., And Montaner, M. (2007) Embedding Emotional Context Inrecommender Systems. In The 20th International Florida Artificial Intelligence Research Society Conference-Flairs, Key West, Florida.

Heckmann, D. (2005). Ubiquitous User Modeling. Phd thesis, Technischen Fakultlaten der Universitlat des Saarlandes, Saarbrucken-Germany.

Heckmann, D. and Kruger, A. (2003). A user modeling markup language (UserML) for ubiquitous computing. In 8th International Conference on User Modeling, LNAI 2702, page 393-397, Johnstown, PA, USA. Springer, Berlin Heidelberg.

Herlocker, Jonathan L. (2000) Understanding And Improving Automated Collaborative Filtering Systems. Tese - Universidade De Minnesota, Minnesota.

Hu,R and Pu,P. (2009a). A comparative user study on rating vs. personality quiz based preference elicitation methods. In Proceedings of the 14th international conference on Intelligent user interfaces (IUI '09). ACM, New York, NY, USA, 367-372.

Hu,R and Pu,P. (2009). Acceptance issues of personality-based recommender systems. In Proceedings of the third ACM conference on Recommender systems (RecSys '09). ACM, New York, NY, USA, 221-224. DOI=10.1145/1639714.1639753 http://doi.acm.org/10.1145/163971.1639753

Hussain, M. S.; Calvo, R. A. (2009). A Framework for Multimodal Affect Recognition. Learning Systems Group, DECE, University of Sydney.

Jensen, C.; Davis, J. and Farnham, S. (2002). Finding others online: reputation systems for social online spaces. In CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 447-454, New York,NY, USA.ACM.

John, Oliver P. E Srivastava, Sanjay. (1999) The Big Five Trait Taxonomy: History, Measurement, And Theoretical Perspectives. In: Handbook Of Personality: Theory And Research, P. 102–138. 1999. Nova Iorque.

Johnson, J.A. (2000) Web-based personality assessment. In 71st Annual Meeting of the Eastern Psychological Association, Baltimore, USA. (Available at http://www.personal.psu.edu/ j5j/vita.html).

John A. Johnson. (2001) Screening massively large data sets for nonresponsiveness in web-based personality inventories. Invited talk to the joint Bielefeld-Groningen Personality Research Group, University of Groningen, The Netherlands. (Available at http://www.personal.psu.edu/ faculty/j/5/j5j/papers/screening.html).

Johnson, J.A. (2005) Ascertaining the validity of individual protocols from web-based personality inventories. Journal of research in personality, 39(1):103–129.

Josang, A., Ismail, R. and Boyd, C. (2007). A survey of trust and reputation systems for online service provision. Decision Support Systems, 43(2):618–644.

Khan, I. A.; Brinkman, W. P.; Fine, N.; Hierons, R. M. (2008) Measuring Personality from Keyboard and Mouse Use, In: European Conference in Cognitive Ergonomics, Madeira, Portugal.

Kobsa, A. (2007). Generic user modeling systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, The Adaptive Web, volume 4321 of Lecture Notes in Computer Science, chapter 4, pages 136–154. Springer Verlag.

Konstan, J. A.; Miller, B. N., Maltz, D.; Herlocker, J. L.; Gordon, L. R. and Riedl, J. (1997). Grouplens: applying collaborative filtering to usenet news. Communications of the ACM, 40(3):77-87.

Lisetti, Christine L. (2002). Personality, affect and emotion taxonomy for socially intelligent agents. In Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference, pages 397–401. AAAI Press.

Mairesse, F.; Walker, M..(2006) Automatic recognition of personality in conversation. In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL), New York City.

Mairesse, F.; Walker, M..(2008) A personality-based framework for utterance generation in dialogue applications. In Proceedings of the AAAI Spring Symposium on Emotion, Personality and Social Behavior.

Mairesse, F., Walker, M.A., Mehl, M.R., Moore, R.K. (2007) Using Linguistic Cues for the Automatic Recognition of Personality in Conversation and Text. Journal of Artificial Intelligence Research 30, p. 457-500.

Mahout. Apache Mahout. (2011) Disponível Em: <Http://Mahout.Apache.Org>. Acesso Em: 03 Mar.

McDonald, David W. (2003). Recommending collaboration with social networks: a comparative evaluation. In CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 593–600, New York, NY, USA. ACM.

Mead, G. H. (1934). Mind, Self, and Society, volume 1. Univeristy of Chicago,Chicago, charles w. morris edition.

Nunes, M. A. S. N. ; Cerri, Stefano A. ; Blanc, N.(2008) . Improving Recommendations By Using Personality Traits In. In: International Conference On Knowledge Management-In: International Conference On Knowledge Management-I.Know08, 2008, Graz-Austria. International Conference On Knowledge Management-I.Know08. V. 1. P. 92-100.

Nunes, M. A. S. N. (2009) Recommender Systems Based On Personality Traits:Could Human Psychological Aspects Influence The Computer Decision-Making Process?1. Ed. Berlin: Vdm Verlag Dr. Müller. V.1. 140p.

Nunes, M. A. S. N. ; Aranha, C. N. (2009) .Tendências à Tomada de Decisão computacional . In: W3C, 2009, São Paulo. W3C.

Nunes, Maria A. S. N. Et Al. (2010) Computação Afetiva E Sua Influência Na Personalização De Ambientes Educacionais: Gerando Equipes Compatíveis Para Uso Em Avas Na Ead. In: Glaucio José Couri Machado. (Org.). Educação E Ciberespaço: Estudos, Propostas E Desafios. Aracaju: Virtus Editora, V. 1, P. 308-347.

Nunes, M. A. S. N. ; Bezerra, J. S. ; Oliveira, A. (2010b) Estendendo O Conhecimento afetivo da EmotionML. In: IHC, 2010, Belo Horizonte. IHC2010. Porto Alegre : SBC.

Paiva, A. and Self, J.A. (1995). Tagus - a user and leamer modelling workbench. User Model. User-Adapt. Interact., 4(3):197–226.

Perugini, Saverio; Gonçalves, Marcos André and Fox, Edward A. (2004). Recommender systems research: A connection-centric survey. Journal of Intelligent Information Systems, 23(2):107–143.

Pianesi, F. ; Mana, N.; Cappelletti, A. ; Lepri, B.; Zancanaro, M..(2008) Multimodal recognition of personality traits in social interactions. in Proceedings of the 10th International Conference on Multimodal Interfaces: Special Session on Social Signal Processing, 2008, pp. 53–60.

Picard, R. W. (1997) Affective Computing. Mit Press, Cambridge, Ma, Usa.

Pina, E. da C.; Nunes, M.A.S.N. (2011) Os Rumos Da Pesquisa Científica Em Computação Afetiva. Relatório Técnico- Notas de Computação Afetiva. Mestrado-PROCC-Universidade Federal de Sergipe.

Poo, D., Chng, B. and Goh, J.M. (2003). A hybrid approach for user profiling. In HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) Washington, DC, USA.IEEE Computer Society.

Porto, S. M.; Costa, S. W.; Nunes, M. A. S. N; Matos, L. N. (2011) Desenvolvimento de Metodologias de Extração de Perfil Psicológico de Usuário para Aplicação em Sistemas de Recomendação Objetivando Personalização de Produtos e Serviços em E-Commerce. Relatório Técnico de Pesquisa. Universidade Federal de Sergipe.

Rabelo, R. A. C.; Nunes, M.A.S.N. (2011) Um Estudo Sobre Modelos E Frameworks De Reconhecimento De Personalidade Em Diálogos. Relatório Técnico- Notas de Computação Afetiva. Mestrado-PROCC-Universidade Federal de Sergipe.

Recio-Garcia, J. A.; Jimenez-Diaz, G.; Sanchez-Ruiz, A. A. and Diaz-Agudo. B. (2009). Personality aware recommendations to groups. In Proceedings of the third ACM conference on Recommender systems (RecSys '09). ACM, New York, NY, USA, 325-328.

Reeves, B. Nass, C. (1996) The Media Equation: How People Treat Computers, Television, And New Media Like Real People And Places. Cambridge University Press, New York, Ny, Usa.

Rein, G. L. (2005). Reputation information systems: A reference model. In HICSS'05: Proceedings of the Proceedings of the 38th Annual Hawaii InternationalConference on System Sciences (HICSS'05) - Track 1, page 26, Washington,DC, USA. IEEE Computer Society.

Resnick, P.; Zeckhauser, R.; Swanson, J. and Lockwood, K.. (2006). The value of reputation on ebay: A controlled experiment. Experimental Economics,9(2):79-101.

Resnick, P.; Kuwabara, K.; Zeckhauser, R. and Friedman, E. (2000). Reputation systems. Commun. ACM, 43(12):45-48.

Resnick, Paul; Varian, Hal R. (1997) Recommender Systems. In: Communications Of The Acm. P. 56-58, Nova Iorque.

Riedl, J. Et Al. (1997) Grouplens: Applying Collaborative Filtering To Usenet News. Communications Of The Acm, New York, V.40, N.3, Pp. 77-87, Mar.

Rousseau, B., Browne, P. , Malone, P. Foster, P. and Mendis, V. (2004). Personalised resource discovery searching over multiple repository types: Using user and information provider profiling. In ICEIS (5), pages 35–43.

Salton, G., McGILL J. M. (1983) Introduction to Modern Information Retrieval. McGraw Hill, New York, USA.

Sandvig, J. J., Mobasher, Bamshad E Burke, Robin. (2007) Robustness Of Collaborative Recommendation Based On Association Rule Mining. In: Acm Conference On Recommender Systems. P. 105-112.

Schafer, J. B.; Konstan, J. and Riedl, J. (1999). Recommender systems in e-commerce. In EC '99: Proceedings of the 1st ACM conference on Electronic commerce, pages 158-166, New York, NY, USA. ACM.

Schafer, J. B.; Konstan, J. and Riedl, J. (2001). E-commerce recommendation applications. Data Mining Knowledge Discovering, 5(1-2):115-153.

Schultz, D. (1990) Theories of Personality. Brooks/Cole, forth edition.

Simon, H.A. (1983) Reason In Human Affairs. Stanford University Press, California.

Thagard, Paul. (2006) Hot Thought: Machanisms And Applications Of Emotional Cognition.A Bradford Book- Mit Press, Cambridge, Ma, Usa.

Tkalčič, Marko Et. Al. (2009) Personality Based User Similarity Measure For A Collaborative Recommender System. In: The 5th Workshop On Emotion In Human-Computer Interaction, P. 279-311, Cambridge.

Tkalčič, M.; Burnik, U.; Košir, A. (2010) Using affective parameters in a content-based recommender system for images. User Modeling and User-Adapted Interaction: The Journal of Personalization Research, Volume 20, Number 4, pages: 279-311.

Trappl, Robert; Payr, Sabine And Petta, Paolo Editors. (2003) Emotions In Humans And Artifacts. Mit Press, Cambridge, Ma, Usa.

Trevisan, Luiz. (2011) Tv Plus: Um Modelo De Sistema De Recomendação De Programas Para Tv Digital. Trabalho De Conclusão De Curso (Graduação Em Sistemas De Informação) - Universidade Do Vale Do Rio Dos Sinos, São Leopoldo, RS.

Salton, G., Mcgill J. M. (1983) Introduction To Modern Information Retrieval. Mcgraw Hill, New York.

Soldz, Stephen E Vaillant, George E. (1998) The Big Five Personality Traits And The Life Course: A 45 Years Longitudinal Study. In: Journal Of Research In Personality, P. 208-232.

W3C 2010a. Emotion Markup Language (EmotionML) 1.0 W3C - Working Draft 29 October 2009. Disponível em: http://www.w3.org/TR/2009/WD-emotionml-20091029/. Acesso em 08/06/2010.

W3C 2010b. SOAP - W3C Recommendation. http://www.w3.org/TR/soap/ . Acesso em 01/07/2010.

W3C 2010c. EMMA: Extensible MultiModal Annotation markup language version 1.0. http://www.w3.org/TR/emma/ . Acesso em 01/07/2010.

W3C 2010d. XML Specification. WorldWideWeb Consortium. Disponível em http://www.w3c.org. Acesso em 01/07/2010.