TÓPICOS EM SISTEMAS COLABORATIVOS, INTERATIVOS, MULTIMÍDIA, WEB E BANCOS DE DADOS

I EDIÇÃO DOS MINICURSOS CONJUNTOS DO SBSC, WebMedia, IHC E SBBD

VII Simpósio Brasileiro de Sistemas Colaborativos (SBSC)

XVI Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia)

IX Simpósio de Fatores Humanos em Sistemas Computacionais (IHC)

XXV Simpósio Brasileiro de Bancos de Dados (SBBD)

05 a 08 de outubro de 2010

Belo Horizonte – Minas Gerais

TÓPICOS EM SISTEMAS COLABORATIVOS, INTERATIVOS, MULTIMÍDIA, WEB E BANCOS DE DADOS

Editora

Sociedade Brasileira de Computação

Organizadores

Adriano C. Machado Pereira

Gisele Lobo Pappa

Marco Winckler

Roberta Lima Gomes

Realização

Universidade Federal de Minas Gerais – UFMG

Promoção

Sociedade Brasileira de Computação

Tópicos em sistemas colaborativos, interativos, multimídia, Web e bancos de dados.

T 673 Adriano C. Machado Pereira , Gisele Lobo Pappa; Marco Winckler, Roberta Lima Gomes / — Belo Horizonte: SBC, 2010.
276p.

1ª edição dos minicursos conjuntos do SBSC, WebMedia, IHC, SBBD. VII Simpósio Brasileiro de Sistemas Colaborativos – SBSC; XVI Simpósio Brasileiro de Sistemas Multimídia e Web- WebMedia; IX Simpósio de Fatores Humanos em Sistemas Computacionais – IHC; XXV Simpósio Brasileiro de Bancos de Dados – SBBD
Realização: Universidade Federal de Minas Gerais,UFMG
ISBN: 85-766-9249-X

1. Sistemas multimídia - Congressos. 2. World Wide Web (Sistema de recuperação da informação) – Congressos I. Pereira, Adriano C. Machado. II. Pappa, Gisele Lobo. III. Winckler, Marco. IV. Gomes, Roberta Lima. V. Sociedade Brasileira de Computação.

CDU 004.019(063)

Prefácio

Este livro contém os textos didáticos redigidos pelos professores dos minicursos selecionados para a edição 2010 de quatro grandes simpósios científicos promovidos pela Sociedade Brasileira de Computação (SBC), a saber: o *VII Simpósio Brasileiro de Sistemas Colaborativos* (SBSC), o *XVI Simpósio Brasileiro de Sistemas Multimídia e Web* (WebMedia), o *IX Simpósio de Fatores Humanos em Sistemas Computacionais* (IHC) e o *XXV Simpósio Brasileiro de Bancos de Dados* (SBBD). Estes quatro eventos, que tradicionalmente aconteciam separadamente, em 2010 foram realizados de forma conjunta, fazendo parte da programação de um evento inédito organizado em Belo Horizonte (MG) de 5 à 8 de outubro de 2010. O objetivo deste livro é, além de apoiar a participação dos presentes durante a realização dos minicursos, ampliar o impacto desses minicursos. Este material garantirá um complemento de informação aos participantes e a todos os interessados nos respectivos temas, permitindo um maior desenvolvimento de seus conhecimentos na área.

Os minicursos são eventos de curta duração (3 ou 6 horas) que visam apresentar uma visão geral de um tópico de pesquisa e/ou tecnologia que seja de interesse das comunidades de um determinado evento, de forma prática e didática. Assim, os participantes têm a oportunidade de aprender sobre novos assuntos vinculados à sua área de atuação, podendo igualmente extrair elementos aplicáveis em seus estudos científicos e/ou profissão.

Através de listas de discussão da SBC, foram encaminhadas à comunidade científica brasileira de computação chamadas de propostas de minicursos que cobrissem os tópicos associados aos quatro eventos (SBSC, WebMedia IHC, SBBD), sendo muito bem vindas as propostas multidisciplinares. As propostas passaram por um processo de seleção conduzido por uma comissão formada por um representante de cada um dos eventos, como segue: Roberta Lima Gomes (SBSC), Adriano César Machado Pereira (WebMedia), Marco Winckler (IHC) e Gisele Lobo Pappa (SBBD). Os critérios utilizados durante a seleção levaram em conta a qualidade técnica da contribuição, originalidade e inovação, relevância para os eventos e potencial para atrair participantes de diferentes comunidades. Recebemos 30 contribuições, todas com mérito. Contudo, devido a limitações de espaço físico, apenas 9 minicursos foram selecionados.

Os capítulos a seguir são apresentados na ordem cronológica em que eles ocorrem segundo a programação do evento:

- **Capítulo 1 – *"Avaliação Multidimensional da Usabilidade de Interfaces para Aplicações Móveis e Multimodais"***: propõe uma abordagem multidimensional de avaliação da usabilidade de interfaces para aplicações móveis e multimodais. São consideradas a sondagem da satisfação dos usuários, mensuração do desempenho dos usuários e inspeção de conformidade do produto a padrões. Tais dimensões complementam-se a fim de produzir um diagnóstico de usabilidade, de natureza quantitativa e qualitativa, mais completo. *Duração do minicurso: 6 horas.*
- **Capítulo 2 – *"Redes Sociais Online: Técnicas de Coleta, Abordagens de Medição e Desafios Futuros"***: apresenta uma introdução a redes sociais, oferecendo uma base necessária ao pesquisador que pretende explorar o tema. São apresentadas inicialmente as principais redes sociais online existentes, os diferentes tipos de conteúdo compartilhados nesses sistemas e as principais técnicas e ferramentas para se obter dados sobre redes sociais. O trabalho também discute métricas e tipos de análises utilizadas no estudo dos grafos que formam a topologia das redes sociais. *Duração do minicurso: 6 horas.*

- **Capítulo 3 – *"Desenvolvimento de Aplicações Imperativas para TV Digital no Middleware Ginga com Java"***: trata do desenvolvimento de aplicações Java para TV digital interativa com base no middleware Ginga, padrão do Sistema Brasileiro de TV Digital. O curso procura explorar as funcionalidades dessas soluções através do desenvolvimento passo a passo de várias aplicações-exemplo e com complexidade crescente, explorando cenários de uso simples (por exemplo, enquete) e avançados (por exemplo, comunicação com outros dispositivos e usuários). O minicurso associado envolve atividades práticas. *Duração do minicurso: 6 horas.*
- **Capítulo 4 –** "***Gerenciamento de Dados em Nuvem: Conceitos, Sistemas e Desafios"***: tem por objetivo apresentar os principais conceitos e tecnologias de computação em *nuvem* (*cloud-computing*) com destaque para o gerenciamento de dados e de sistemas. O trabalho termina com uma discussão sobre os desafios e oportunidades na área. *Duração do minicurso: 3 horas.*
- **Capítulo 5 – "*Minerando a Web por meio de Grafos - da teoria às aplicações"***: apresenta uma introdução às técnicas de mineração de grafos, principalmente aqueles oriundos da Web como as redes sociais. São apresentados conceitos básicos, técnicas de mineração de grafos e as principais áreas de aplicação de tais técnicas. *Duração do minicurso: 3 horas.*
- **Capítulo 6 – *"Processo de KDD aplicado à Bioinformática"***: apresenta uma discussão sobre as etapas de construção de um ambiente de descoberta de conhecimento em base de dados (*Knowledge Discovery in Databases* **–** *KDD*). Em especial, são abordados desafios relacionados ao tratamento de dados peculiares e de característica não convencional, como na bioinformática. *Duração do minicurso: 3 horas.*
- **Capítulo 7 – "*An Introduction to Model-based User Interface Development to realize Multimodal and Context-sensitive Applications for Smart Environments*"**: apresenta uma introdução ao processo de desenvolvimento de interfaces de usuários baseado em modelos. Este trabalho aborda o problema sob a perspectiva do desenvolvimento de interfaces para diferentes plataformas, o que inclui adaptação das técnicas de interação multimodal para o dispositivo. Embora o texto seja em Inglês, o minicurso foi apresentado em Português. *Duração do minicurso: 3 horas.*
- **Capítulo 8 – *"Garantia de Segurança em Aplicações: De Requisitos a Padrões Seguros"***: oferece aos participantes uma visão sobre segurança de sistemas e engenharia, focando nos aspectos de segurança dentro do ciclo de desenvolvimento de software. *Duração do minicurso: 3 horas.*
- **Capítulo 9 – *"Desenvolvimento de Gerenciador de Conteúdo com Tecnologia Web 2.0"***: descreve conceitos e práticas relacionadas ao desenvolvimento de gerenciadores de conteúdo (*Content Management Systems – CMS*). Também são apresentadas tecnologias necessárias ao desenvolvimento de aplicações baseadas em CMS, tais como PHP orientada a objetos, com banco de dados MySQL e Ajax com jQuery. O minicurso associado envolve atividades práticas. *Duração do minicurso: 6 horas.*

Os trabalhos aqui reunidos compõem um painel de tópicos representativos dos eventos SBSC, WebMedia, IHC, SBBD. Agradecemos o interesse da comunidade em contribuir com os minicursos e esperamos que os leitores encontrem neste volume informações úteis que lhes auxiliem a progredir nas suas atividades profissionais, sejam elas acadêmicas e/ou industriais.

Adriano C. Machado Pereira, Gisele Lobo Pappa, Marco Winckler e *Roberta Lima Gomes*.

Coordenadores dos Minicursos (WebMedia / SBBD / IHC / SBSC)

Outubro de 2010.

Sumário

Capítulo 1

Avaliação Multidimensional da Usabilidade de Interfaces para Aplicações Móveis e Multimodais

Ana Esther Victor Barbosa, Danilo de Sousa Ferreira, José Eustáquio Rangel de Queiroz

Abstract

This mini-course will provide usability practitioners, HCI students and lecturers with a description of a multidimensional approach for the evaluation of mobile application and multimodal application user interfaces. Each dimension of this approach focus on the problem from a different perspective: (i) the user's perspective, which is expressed as views on the product gathered from an inquiry-based approach; (ii) the specialist's perspective, which is expressed by a specialist analysis of the user performance during the usability evaluation; and (iii) the usability community's perspective, which is expressed by a specialist analysis, based on standards inspection. Since the multidimensional approach is based on qualitative and quantitative usability data, a more reliable and complete product appraisal results from the combination of the above-mentioned perspectives.

Resumo

O presente minicurso visa apresentar uma abordagem multidimensional de avaliação da usabilidade de interfaces para aplicações móveis e multimodais, destinando-se a estudantes, professores e profissionais de usabilidade. Cada uma das dimensões foca os problemas de usabilidade a partir de diferentes perspectivas, a saber: (i) a perspectiva do usuário, expressa mediante a sondagem de sua satisfação no tocante ao produto avaliado; (ii) a perspectiva do especialista, expressa a partir da análise da mensuração do desempenho do usuário ao utilizar o produto avaliado; e (iii) a perspectiva da comunidade de usabilidade, expressa por meio da inspeção da conformidade do produto a padrões. Tais dimensões complementam-se, a fim de produzir um diagnóstico mais confiável e completo da usabilidade do produto avaliado, uma vez que se fundamenta em dados de usabilidade de natureza quantitativa e qualitativa.

1.1 Introdução

Nos últimos anos, com a evolução das tecnologias, constata-se a presença cada vez mais efetiva de sistemas computacionais (aplicações de *software* ou *hardware*) nas atividades cotidianas dos indivíduos. Devido à revolução dos computadores pessoais e à redução dos custos de *hardware*, que tem tornado os sistemas computacionais cada vez mais acessíveis a um número cada vez maior e mais diversificado de usuários, as interfaces usuário-computador tornaram-se entidades muito mais significativas do que costumavam ser, há pouco mais de duas décadas. Enquanto há algum tempo apenas usuários experientes em sistemas computacionais desenvolviam sistemas para uso próprio ou para outros usuários experientes, nos dias atuais os sistemas computacionais vêm se tornando praticamente onipresentes e contemplam cada vez mais o segmento crescente de usuários principiantes, atuantes nas mais diversas áreas do conhecimento humano [Queiroz, 2001].

A *interface usuário-computador*, também denominada *interface com o usuário*, *interface homem-computador* ou simplesmente *interface*, é um fator determinante para a aceitação de produtos interativos, uma vez que proporciona o elo de comunicação entre o usuário e um dado sistema computacional, exercendo influência no modo como o usuário visualiza e entende a funcionalidade de um sistema. Tal fato vincula sua concepção a aspectos do sistema com os quais o usuário interage física e cognitivamente [Barbosa, 2009]. Assim sendo, faz-se necessário que a interface comporte-se de maneira consistente e adequada, não bastando apenas que a funcionalidade esteja correta, pois se a utilização do produto não proporcionar conforto e bem-estar ao usuário, seu desempenho ao usá-lo tenderá a cair, pois a satisfação na realização de uma tarefa é um fator que impacta diretamente na produtividade.

Vários autores (e.g., [Queiroz, 2001], [Tidwell, 2005], [Seffah e Metzker, 2009]) consideram que, no âmbito da qualidade de *software*, a usabilidade tem se tornado uma necessidade competitiva para o sucesso comercial dos produtos de *software*, além de vir sendo incorporada cada vez mais efetivamente aos processos de desenvolvimento, com o propósito de melhorar a qualidade dos produtos.

Deste modo, a avaliação da usabilidade de produtos interativos vem ganhando um espaço cada vez mais bem delineado no domínio da comunicação mediada por computadores, constituindo uma etapa essencial e integral do projeto e desenvolvimento da interação usuário-computador.

1.1.1 Introdução à Usabilidade

Shackel (1991), Nielsen (1993) e a Parte 11 do padrão ISO 9241 (1998a) descreveram visões da usabilidade nas quais levantam questionamentos, em um nível operacional, sobre objetivos de usabilidade e sobre a relação entre a usabilidade, a utilidade, a aceitação de produtos e questões relativas ao processo interativo. A seguir, são apresentadas as visões de Nielsen (1993) e do padrão ISO 9241 (1998a).

Nielsen (1993) considerou que a aceitação de um sistema podia ser mensurada a partir de vários critérios, e.g. aplicabilidade (*usefulness*), confiabilidade, custos, compatibilidade. A aplicabilidade foi associada à utilidade e à **usabilidade** do produto. O autor empregou alguns critérios para esclarecer o conceito de usabilidade, a saber:

- ***Facilidade de aprendizado***: aprender a utilizar um novo sistema deve ser suficientemente fácil para que o usuário comece a usá-lo rapidamente;

- ***Eficiência***: o sistema deve ser eficiente de tal forma que os usuários possam atingir uma alta produtividade;
- ***Facilidade de memorização***: um usuário ocasional do sistema não precisa re-aprender a utilizar o sistema, o sistema precisa ser intuitivo;
- ***Tolerância a erros***: o sistema deve possibilitar ao usuário a fácil recuperação em situações de erros;
- ***Satisfação***: o usuário deve ficar satisfeito com o uso do sistema.

A Parte 11 (Especificações de usabilidade) do padrão internacional ISO 9241 (1998a), *Ergonomic Requirements for Office Work with Visual Display Terminals*, define usabilidade como a *eficácia*, *eficiência* e *satisfação* com as quais usuários específicos atingem metas específicas em ambientes específicos, i.e. em um contexto de uso específico. Tais dimensões foram associadas pela ISO a:

- ***Eficácia***: precisão e completude com a qual os usuários alcançam metas específicas;
- ***Eficiência***: recursos necessários para que os usuários alcancem suas metas com completude e precisão;
- ***Satisfação***: conforto e aceitação do uso do sistema pelos usuários.

A incorporação da usabilidade no desenvolvimento de um produto conduz a uma variedade de benefícios potenciais, a saber [Bias & Mayhew, 1994], [Bosert, 1991], [Nielsen, 1993]: (i) redução tempo de desenvolvimento, devido à diminuição de retrabalhos; (ii) redução nos custos de concepção, pois mudanças são antecipadas; (iii) redução nos custos com manutenção e correção de erros, uma vez que menos erros serão encontrados; (iv) redução no tempo/custos de treinamento; (v) redução do número de chamadas ao suporte; (vi) redução no número de consultas à ajuda (manual ou *help online*); (vii) aumento da satisfação dos usuários, conseqüência do aumento de produtividade (eficiência, eficácia, menor taxa de erros cometidos); e (viii) aumento nas vendas do produto, em virtude do aumento da satisfação.

1.1.2 Métodos de Avaliação de Interfaces Usuário-computador

Avaliação da usabilidade consiste em qualquer estudo de natureza analítica ou empírica de um sistema ou protótipo, cuja meta de avaliação seja fornecer um diagnóstico do produto ou sistema avaliado.

Os processos de avaliação podem ser de natureza ***formativa*** ou ***somativa***. ***Avaliações formativas*** (*formative evaluation*) ocorrem durante o ciclo de desenvolvimento do produto, inclusive durante a fase de projeto. O objetivo dos estudos formativos é identificar aspectos no produto, seja um protótipo ou versões de teste, que podem ser melhorados, i.e. sugerir mudanças a serem incorporadas ao produto final.

Por sua vez, as ***avaliações somativas*** (*summative evaluation*) têm o propósito de fornecer diagnósticos da interface ao término de diferentes etapas de seu desenvolvimento. Este tipo de avaliação geralmente ocorre no fim do processo de desenvolvimento, quando se testa o produto para avaliar se as metas de usabilidade foram atingidas.

Preece *et al.* (2005) identificaram quatro paradigmas centrais de avaliação, a saber: (i) ***métodos rápidos*** (*quick-and-dirty methods*) e ***econômicos*** (*discount methods*); (ii) ***testes de usabilidade***; (iii) ***estudos de campo***; e (iv) ***avaliação preditiva***.

Métodos rápidos e ***econômicos*** envolvem a aquisição, de forma rápida e barata,

de opiniões dos usuários ou consultores sobre o produto, podendo ser adotados em qualquer etapa do ciclo de desenvolvimento.***Testes de usabilidade*** envolvem, por sua vez, a avaliação do desempenho de usuários típicos na realização de tarefas típicas.

Estudos de campo caracterizam-se por serem realizados em ambientes reais ou próximos do real, tendo como objetivo principal maximizar o entendimento acerca de como os usuários agem e de como a tecnologia impacta em suas atividades. Por fim, ***avaliações preditivas*** caracterizam-se pela análise do produto por especialistas, com base em heurísticas ou recomendações de usabilidade, sem o envolvimento de usuários no processo.

Independentemente de sua natureza, os estudos de avaliação da usabilidade têm como objetivo principal analisar um produto sob a perspectiva dos aspectos que impactam diretamente no seu uso. Todavia, é importante considerar tal tarefa pode envolver uma ou mais técnicas adequadas aos objetivos da avaliação.

Diversas iniciativas de categorizar técnicas de avaliação podem ser encontradas na literatura clássica de HCI (*Human-Computer Interaction*). Uma das classificações mais conhecidas é aquela que categoriza os métodos em ***empíricos*** e ***analíticos***.

Os métodos ***empíricos*** caracterizam-se pelo envolvimento dos usuários no processo avaliatório. Tais métodos podem ser informais, e.g. observação de usuários ao utilizar protótipos do sistema, ou formais, e.g. observação em ambiente laboratorial controlado [Rosson & Carroll, 2002].

Os métodos ***analíticos*** fundamentam-se na análise das interfaces por avaliadores (ergonomistas, engenheiros de *software* ou de usabilidade) que investigam aspectos da usabilidade do produto baseado em recomendações, padrões ou heurísticas.

Os métodos e técnicas podem ainda serem enquadrados, segundo Nielsen (1993), como pertencentes a uma das seguintes categorias: (i) ***ensaios de usabilidade*** (*usability testing*) ou (ii) ***inspeções de usabilidade*** (*usability inspections*).

Ensaios de usabilidade consistem basicamente de estudos de um processo interativo usuário-computador específico, em condições "reais" ou "controladas", nos quais especialistas em interfaces coletam dados sobre eventos relacionados com a interação propriamente dita e problemas afins ocorridos durante o uso da aplicação por uma amostra da comunidade usuária [Nielsen, 1993], [Dix, 2003]. Envolvem uma gama de técnicas que se diferenciam segundo o grau de especialidade necessário, a formalidade do procedimento e os custos associados [Dix, 2003].

Em geral, ***inspeções de usabilidade*** são estratégias fundamentadas na análise e julgamento de projetos por avaliadores (ergonomistas, engenheiros de *software*, engenheiros de usabilidade), que investigam aspectos relativos à usabilidade segundo um conjunto de critérios, recomendações, normas ou heurísticas.

As principais estratégias adotadas em ensaios de usabilidade são sintetizadas no Quadro 1, enquanto as principais estratégias que se enquadram na categoria inspeções da usabilidade são sumariadas no Quadro 2. Técnicas adaptativas ou inovadoras vêm sendo pesquisadas e continuamente propostas, uma vez que o surgimento de novas tecnologias, de *hardware* ou *software*, têm imposto novas formas de interação com o usuário.

Quadro 1 – Estratégias usualmente adotadas em Ensaios de Usabilidade.

Métodos	Variantes	Descrição	Observações
Observações (*Observation*)	**Observação Cooperativa do Usuário**	Consiste na observação, por uma equipe de especialistas, de usuários interagindo com o produto-alvo.	Registros em áudio e vídeo são utilizados para coleta de dados.
	Observação em Ambiente Controlado (laboratorial)	Usuários de teste são observados, por especialistas, durante a execução de tarefas pré-definidas. Os avaliadores podem observá-los na mesma sala ou através de uma superfície refletora unidirecional.	Geralmente, registros em áudio e vídeo são utilizados como instrumentos de coleta de dados.
	Observação em Ambiente Natural (de Campo)	Consiste na observação de usuários realizando tarefas em seu ambiente natural (e.g. ambiente de trabalho).	
Questionários (*Questionnaires*)	**Questionário Conduzido por um Entrevistador**	Entrevistadores treinados conduzem tais questionários, permitindo um melhor controle no processo de aquisição de dados e maior interação com os usuários.	Os questionários podem ser *abertos* ou *fechados*.
	Questionário Autodirigido	Os próprios usuários fazem o preenchimento do questionário.	
Entrevistas (*Interviews*)	**Entrevista Aberta Padronizada**	Aplicação de um mesmo conjunto de questões para uma amostra de usuários.	O registro de áudio, com posterior transcrição, pode ser feito em substituição às anotações.
	Entrevista Estruturada ou Guiada	Entrevistador conduz um questionamento mais formal para os participantes sobre o tema em questão.	
	Entrevista Não-estruturada ou Informal	Entrevistador adapta as questões, a fim de respeitar as diferenças individuais e/ou acompanhar mudanças comportamentais.	
Verbalização de Procedimentos (*Thinking aloud*)		Consiste na verbalização, por parte do usuário de teste, de todos os procedimentos, idéias, encadeamentos lógicos e opiniões indispensáveis à conclusão da tarefa.	
Interação Construtiva (*Constructive interaction*)		Consiste em dois usuários operando conjuntamente o produto-alvo da avaliação. É denominada também de *aprendizagem por compartilhamento de descobertas* (*codiscovery learning*).	
Ensaio Retrospectivo (*Retrospective testing*)	**Ensaio Retrospectivo Clássico**	Consiste no registro em vídeo da sessão de teste e na análise posterior do transcurso da sessão **sem** a presença do usuário.	Câmeras de vídeo devem registrar pelo menos eventos ocorridos em tela, expressões faciais e mãos do usuário de teste.
	Covisualização	Consiste no registro em vídeo da sessão de teste e na análise posterior do transcurso da sessão **com** a presença do usuário.	
Captura Automática (*Automatic logging*)		Consiste na monitoração e coleta automática de informações relativas ao uso do produto sob avaliação.	Vários dados podem ser coletados, e.g. ações com o *mouse*, teclas acionadas.
Discussões em Grupo (*Focus groups*)		Consiste na reunião, mediada por um moderador, de seis a nove usuários. Utilizada para coletar informações sobre as necessidades e opiniões dos usuários.	
Ensaio de Usabilidade Remoto (*Remote usability testing*)		Consiste num ensaio de usabilidade conduzido em situações nas quais avaliadores e usuários de teste encontram-se fisicamente separados.	Há várias opções para conduzir testes remotos, e.g. vídeo-conferência, relatórios *on-line*.

Fonte: Adaptado de [Queiroz, 2001].

Quadro 2 - Estratégias usualmente adotadas em Inspeções de Usabilidade.

Método	Variantes	Descrição
Revisão Sistemática (*Walkthrough*)	**Revisão Sistemática Tradicional**	Consiste na análise estruturada de um produto por uma equipe de especialistas.
	Revisão Sistemática Cognitiva (*Cognitive walkthrough*)	Consiste na avaliação, por um grupo de especialistas, das diferenças existentes entre as metas e expectativas dos usuários e os procedimentos requisitados pela aplicação.
	Revisão Sistemática Cognitiva Automatizada (*Automated cognitive walkthrough*)	As várias respostas e questões relativas às ações do usuário são coletadas automaticamente.
	Revisão Sistemática Pluralista de Usabilidade (*Pluralistic usability walkthrough*)	Revisão na qual tanto a população usuária, como os projetistas do produto e os especialistas em fatores humanos agem de forma colaborativa.
	Jogthrough	Revisão conduzida em um "ritmo mais acelerado", i.e. tal técnica apresenta-se como uma versão abreviada da Revisão Sistemática Tradicional.
Inspeção (*Inspection*)	**Inspeções baseadas em Diretrizes de Projeto**	Diretrizes de projeto são sugestões e recomendações técnicas que sumariam princípios bem conhecidos aplicáveis a projetos de interface de usuário.
	Inspeções baseadas em Guias de Estilo	Guias de estilo são documentos que sumariam padrões industriais e contém informações prescritivas, ao invés de sugestivas.
	Inspeções baseadas em Padrões	Os padrões especificam como as interfaces deveriam ser apresentadas ao usuário. Representam um consenso.
	Avaliações Heurísticas	Consiste num estudo conduzido por especialistas a partir de heurísticas que guiam as atividades dos avaliadores.

Fonte: Adaptado de [Queiroz, 2001].

1.1.3 Caracterização dos Dispositivos Móveis e Multimodais

Nos últimos anos, o cotidiano de indivíduos do mundo inteiro vem exigindo cada vez mais a incorporação de aplicações e/ou dispositivos computacionais às mais diversas atividades, os quais são utilizados como ferramentas de trabalho, pesquisa, compra, entretenimento, comunicação, etc.

Os avanços no desenvolvimento de tecnologias de *hardware* tornaram os computadores cada vez mais complexos e capazes de proporcionar soluções a uma ampla variedade de problemas, melhorando a qualidade de vida dos seres humanos. Todavia, a mesma tecnologia que simplifica a vida, provendo um maior número de funcionalidades em um objeto, também pode complicá-la, tornando essa tecnologia muito mais difícil de aprender e usar, o que exige, muitas vezes, dos usuários a mudança de hábitos e a adaptação aos novos métodos de trabalho.

Segundo Rosson e Carroll (2002), as tecnologias atuais têm evoluído na direção da computação ubíqua[1] (e.g. *handhelds*, *palmtops* e celulares). Estes dispositivos integram a computação móvel, i.e. atividades fundamentadas no uso de computadores (*computer-based activities*), as quais podem ser executadas quando os usuários não estão nos seus locais de trabalho.

Diante da evolução dos dispositivos computacionais e das tecnologias de um conceito estático[2] para um contexto móvel, torna-se perceptível uma maior adoção de dispositivos móveis do que dispositivos não-móveis, e.g. os celulares, que têm se tornado bastante populares. Além do mercado de telefonia celular e dos seus serviços, constata-se um aumento considerável na comercialização de PDA (*Personal Digital Assistants*) e *smartphones*[3].

Ao avaliar a diversidade de dispositivos móveis disponíveis hoje no mercado, constata-se o emprego cada vez maior de interfaces multimodais em tais dispositivos (MUI – *Multimodal User Interfaces*). As MUI são uma evolução das interfaces gráficas com o usuário (*Graphical User Interfaces*) no sentido de melhorar o processo interativo homem-computador (*Human-Computer Interaction – HCI*) [Oviatt, 2003].

A demanda crescente por sistemas com elevado grau de usabilidade e que não interfiram significativamente na maneira como as pessoas realizam suas tarefas, i.e., que sejam minimamente intrusivos, resultaram no desenvolvimento de interfaces computacionais que suportam formas mais naturais de comunicação, por meio da fala, visão, toque, gestos manuais, etc. [Inácio Jr., 2007]. As MUI surgiram para tirar proveito dos diferentes modos de interação existentes, possibilitando o aumento da largura de banda na comunicação a partir do processamento simultâneo de diferentes tipos de entrada [Dix *et al*., 2003], [Bernhaupt *et al*., 2007].

Ao contrário das habituais GUI, caracterizadas pelo uso do teclado, *mouse* e janelas, as MUI processam um conjunto de modalidades de entrada do usuário, tais como reconhecimento de voz, gestos, escrita manual, movimentos de cabeça ou *mouse*, rastreamento de movimentos oculares (*eye-tracking*), etc., de uma maneira integrada e

---

[1] Em 1991, Weiser [Weiser, 1991] usou o termo *computação ubíqua* para descrever uma visão do futuro, na qual os computadores estariam integrados ao mundo que nos cerca, auxiliando-nos nas tarefas diárias.

[2] Sem mobilidade, sem portabilidade.

[3] Dispositivos que agregam funcionalidades de *PDA* e de telefones celulares.

coordenada com a saída multimídia do sistema [Wahlster, 2006], [Oviatt & Cohen, 2000 apud Inácio Jr., 2007]. Tais modalidades permitem aos usuários a interação com os computadores, a partir de diferentes modos ou canais de comunicação.

As MUI também possibilitam aos usuários com diferentes níveis de habilidades a escolha do modo de interação mais adequado às suas necessidades e preferências, proporcionando uma maneira mais natural e flexível para a realização de tarefas, permitindo maior interação de usuários humanos com computadores e possibilitando a escolha de qual modalidade usar, como também a combinação de diferentes modalidades de interação [Maybury, 2001], [Inácio Jr., 2007], [Stanciulescu *et al*., 2007], [Blumendorf *et al*., 2008]. A integração de modalidades resulta em uma comunicação mais eficiente (execução de tarefas de forma mais rápida e com menor esforço) e eficaz (seleção de tarefas e diálogos mais direcionados ao contexto do usuário) [Maybury, 2001]. O objetivo das MUI é assemelhar a interação homem-computador àquela segundo a qual os humanos interagem entre si e com o ambiente [Dix *et al*., 2003].

Uma vez que o desenvolvimento de MUI não é *per si* um indicativo da garantia de sua usabilidade, a simples adição de novas modalidades não garante o aumento da qualidade de uma interface, sendo necessário que haja uma integração adequada para que a nova modalidade agregue algum valor ao processo interativo.

I Dispositivos Móveis

Algumas divergências são encontradas quanto às terminologias e classificações adotadas por diferentes pesquisadores em relação aos dispositivos móveis. Portanto, o objetivo desta subseção é caracterizar precisamente a categoria de dispositivos móveis considerada no escopo deste mini-curso.

Móvel é a habilidade de estar em movimento, é uma característica de algo que pode ser utilizado em movimento. Portanto, a característica móvel pode ser empregada em um dispositivo computacional se tal dispositivo pode ser facilmente transportado de um local para outro enquanto o usuário interage com ele. Ou seja, a mobilidade é um atributo tanto do dispositivo quanto do usuário.

Dispositivos que podem ser transportados para qualquer local, mas exigem que o usuário permaneça parado (sem movimentação) durante a interação são classificados como **portáteis** (i.e. de fácil transporte) [Gorlenko & Merrick, 2003]. Em muitos casos, conforme Mallick (2003), os termos **móvel** e **sem fio** são utilizados de forma inversa. **Sem fio** (*wireless*) refere-se à transmissão de dados através de ondas de rádio.

Na Figura 1 é ilustrado o relacionamento entre a conceituação dos termos **móvel**, **sem fio** e **portátil**. Há dispositivos (i) móveis sem acesso à transmissão de dados, seja com fio (*wired*) ou sem fio (*wireless*) (e.g. jogos eletrônicos, câmeras e reprodutores de MP3); (ii) apenas portáteis, mas com transmissão de dados sem fio (e.g. *laptop* com acesso *wireless* à Internet); (iii) *desktop* (dispositivos estacionários) com acesso sem fio. A classificação adotada neste documento baseia-se em dois tipos de categorizações: primeiramente, serão categorizados os dispositivos de computação pessoal; e, em seguida, no âmbito da categoria alvo de computação pessoal, serão apresentados os tipos de dispositivos que se enquadram nesta categoria.

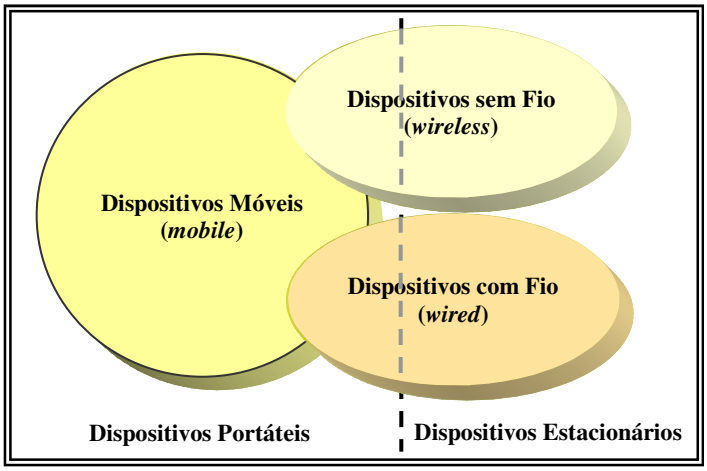


Figura 1 - Relacionamento dos termos *móvel*, *portátil* e *sem fio*.

Na Figura 2, ilustra-se a classificação da computação pessoal, concebida por Gorlenko & Merrick (2003), baseados na classificação feita por Weiss (2002), através de 5 categorias distintas de dispositivos.

Observa-se, a partir da Figura 2, que o tamanho do dispositivo diminui à medida que o grau de mobilidade aumenta. Adicionalmente, conforme ressaltado por Weiss (2002), pode-se perceber que a fronteira entre as categorias é tênue. Esta interseção entre as categorias mostra que algumas dificuldades podem ser encontradas ao tentar classificar determinados dispositivos em apenas uma das categorias. As categorias identificadas na Figura 2 foram diferenciadas conforme aspectos de dimensão, peso e grau de mobilidade. Este último aspecto permite analisar a influência de tal aspecto no modo de interação.

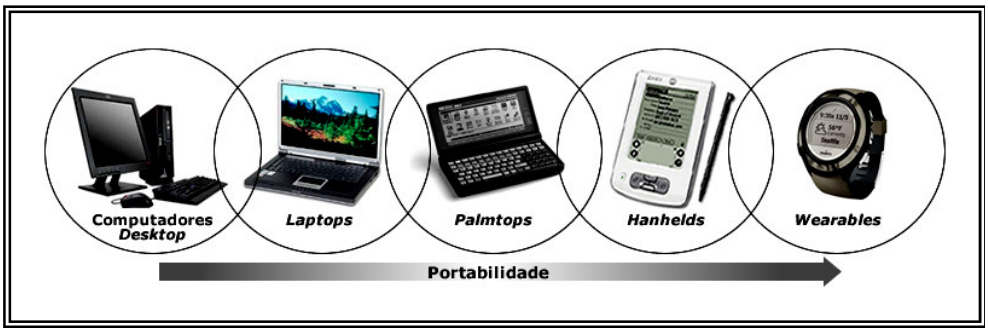


Figura 2 - Classificação da computação pessoal.

Fonte: [Gorlenko & Merrick, 2003].

No Quadro 3 são sumariadas as principais características das cinco categorias de dispositivos supracitadas, quanto ao grau de mobilidade, o tamanho e peso, o modo de interação e a modularidade de seus componentes. Os dispositivos-alvos desta pesquisa, cujo objetivo é propor uma metodologia de avaliação da usabilidade, são aqueles enquadrados na categoria *handheld*. Ao longo deste documento, o emprego do termo dispositivo móvel se referirá aos dispositivos contemplados na categoria *handheld*, a qual engloba várias famílias de dispositivos, a saber: telefones celulares, *pagers*, PDAs (*Personal Digital Assistants*), *smartphones* e *tablets*. No Quadro 4, sumariam-se as principais características dos dispositivos classificados neste documento como dispositivos móveis.

Quadro 3 – Classificação dos dispositivos computacionais.

Categoria do Dispositivo	Tamanho Físico	Grau de Mobilidade	Modo de Interação	Modularidade
Desktop	Grande	Fixo	Estacionário	Completamente modularizado
Laptop	Médio	Portátil	Estacionário	Módulo único
Palmtop	Pequeno	Portátil	Predominantemente estacionário	Módulo único
Handheld	Médio/Pequeno	Móvel	Móvel	Módulo único
Wearable	Pequeno	Móvel	Móvel	Completamente modularizado

Fonte: [Gorlenko & Merrick, 2003].

Quadro 4 – Diferenças entre telefones celulares, *pagers*, *PDAs*, *smartphones* e *tablets*.

Dispositivo	Principal Função	Mecanismo de Entrada	Dimensões da Tela	Porte
Telefone Celular	Ligações telefônicas	Teclado Numérico	Muito pequena. Geralmente, 12 caracteres por 4 linhas	Pequeno
Pager	Envio de mensagens textuais	Teclado QWERTY	Varia de 1 linha por 16 caracteres até 160 x 160 *pixels*	Pequeno
PDA	Armazenamento e recuperação de informação	Tela sensível ao toque	160 x 160 pixels é comum, mas são geralmente maiores	Médio
Smartphone	Ligações telefônicas; Armazenamento e recuperação de informação	Teclado Numérico; Teclado para entrada de dados textuais	De 640 x 200 a 320 x 240 *pixels*	Médio
Tablet	Armazenamento e recuperação da informação	Tela sensível ao toque	Até 1024 x 768 *pixels*	Grande

Fonte: [Weiss, 2002].

II Interfaces Multimodais

As interfaces com o usuário são tradicionalmente unimodais, i.e., oferecem apenas uma modalidade de entrada ao usuário (e.g., teclado e *mouse*). Por outro lado, as MUI integram informações de "vários modos", oferecendo "múltiplas modalidades de interação" (e.g., fala, gestos, escrita manual, movimentos de cabeça ou *mouse*, rastreamento de movimentos oculares), cada uma das quais pode corresponder a um dos cinco sentidos físicos utilizados na comunicação e à seleção dos pares adequados modalidade/técnica de interação (e.g., fala/reconhecimento de voz, escrita manual/manipulação direta (interação via caneta)).

As MUI processam um conjunto de modalidades de entrada do usuário de uma maneira integrada e sincronizada com a saída multimídia do sistema [Oviatt & Cohen, 2000 apud Inácio Jr., 2007].

Apesar de multimodalidade está geralmente associada com a possibilidade de o usuário utilizar vários dispositivos de entrada, multimodalidade refere-se também à saída de um processo interativo usuário-sistema. Um aspecto específico das MUI é que as técnicas de interação, os dispositivos de entrada e saída e os canais sensoriais estão intimamente relacionados [Bernhaupt *et al*., 2007]. Uma técnica de interação pode envolver um ou mais dispositivos de entrada ou combinações de dispositivos, e.g., uma tela sensível ao toque pode ser utilizada como um dispositivo de entrada para várias técnicas de interação, tais como reconhecimento de escrita, manipulação direta, teclado virtual e interação por gestos.

1.1.4 Usabilidade de Dispositivos Móveis e Multimodais

Em se tratando do mercado de dispositivos móveis, principalmente de telefones celulares, a usabilidade pode tornar-se um importante diferencial competitivo.

A usabilidade inadequada de um dispositivo móvel pode implicar várias conseqüências no tocante ao usuário, dentre as quais: (i) a incapacidade de completar tarefas de interesse; (ii) a incapacidade de utilizar serviços; e (iii) a insatisfação com o produto. Infelizmente, nem sempre os usuários podem distinguir se o problema é causado pela interface do produto, pela conexão de rede ou pelo serviço utilizado. Para as operadoras de telefonia e para os provedores de serviço, falhas na usabilidade levam à redução drástica do uso dos serviços. Para os fabricantes, isto vem a culminar com a redução significativa das vendas dos dispositivos [Ketola; Röykkee, 2002].

Há uma grande variedade de aspectos em telefones celulares carentes de investigação, tais como: facilidades de reconhecimento de voz [Zhu, 2004], estrutura de menus [St. Amant *et al*., 2004]; [Klockar *et al*., 2003], recomendações e diretrizes para o conteúdo da Internet Móvel [Duda *et al*., 2001]; [Grant, 2004] e utilização do espaço reduzido mais eficientemente [Kamba *et al*., 1996].

Kjeldskov (2002) identificou três obstáculos principais de usabilidade que podem ser encontrados no projeto de aplicações para dispositivos móveis, a saber: (i) visor bastante reduzido; (ii) mecanismos de entrada de dados limitados; e (iii) contextos de uso dinâmicos.

Algumas pesquisas (vide e.g. Croasmun (2004)) sugerem que a maioria dos celulares disponíveis no mercado tem teclas de tamanho adequado ao dedo de uma criança de 5 anos de idade. E, com a miniaturização cada vez maior destes dispositivos,

outros mecanismos de entrada de dados precisam ser fornecidos aos usuários.

St. Amant *et al*. (2004) relataram que o projeto de menus hierárquicos para telefones celulares é um problema não trivial devido a alguns fatores: (i) necessidade de ações de seleção discreta, sob a forma de ativação das teclas, para o movimento de um item de menu para outro, uma vez que a maioria dos telefones celulares carece de facilidades de seleção mais direta (e.g. um mouse ou uma tela sensível ao toque); (ii) dimensões reduzidas das telas de celulares, o que permite a visualização de somente um item de menu por vez; e (iii) pouca padronização em nível de hardware de suporte à navegação via menus para telefones celulares quando comparada às iniciativas de padronização para computadores *desktop*.

No que diz respeito às interfaces multimodais, pode-se destacar como principal vantagem oriunda da integração de modalidades, o aumento de eficiência no tratamento de erros, de acordo com os erros provenientes do usuário ou do sistema. Oviatt *et al*. (2000) (apud Inácio Jr., 2007) descrevem que algumas das razões para que tal vantagem exista são, dentre outras: (i) a escolha pelo usuário do modo de interação menos propenso a erros, em função da natureza da tarefa; (ii) a tendência do usuário de alterar o modo de interação após o cometimento de erros, o que favorece o aprendizado dos diferentes modos de interação disponíveis; (iii) a simplificação da linguagem do usuário ao interagir multimodalmente, o que reduz a complexidade do processamento de linguagem natural e, portanto, diminui erros de reconhecimento; e (iv) o suporte de arquiteturas multimodais à desambiguação mútua das entradas, nas quais a informação semântica de uma modalidade funciona como entrada parcial para a desambiguação da outra.

As novas modalidades de interação vêm se tornando essenciais para suportar os estilos atuais de interação e proporcionar experiências mais ricas para o usuário. É neste ponto que as interfaces multimodais têm se sobressaído, pois possibilitam um aumento na eficiência com que a informação pode ser obtida/gerada, além de permitir entradas paralelas da maneira que o usuário considerar mais conveniente. A usabilidade e a acessibilidade aprimorada dos sistemas multimodais os torna passíveis de acomodar um grande número de tarefas nos mais diversos ambientes [Inácio Jr., 2007].

Estudos de Dillon *et al*. (1990) e Kjeldskov & Stage (2004) revelaram que a simples adição de novas modalidades não garante o aumento da qualidade de uma interface, ou seja, que a interação será mais fácil e eficiente, pois as MUI mal projetadas não aportam ganho, se comparadas a interfaces com o usuário que adotem uma modalidade de interação convencional (e.g., teclado e *mouse*).

Para determinar a contribuição das modalidades na interação homem-computador, muitos estudos empíricos têm sido realizados em termos de investigar *como*: (i) a usabilidade e a aceitação do usuário têm sido influenciadas por novos dispositivos e técnicas de interação [Hinckley *et al*., 1998]; [Bowman *et al*., 2002]; [Nedel *et al*., 2003]; [Poupyrev *et al*., 1998]; (ii) a usabilidade percebida está sendo influenciada pela natureza das tarefas executadas [Dybkjær *et al*., 2004b]; [Jöst *et al*., 2005] e pelo contexto de uso (e.g., condições laboratoriais versus campo, mobilidade versus estacionariedade) [Baille & Schatz, 2005]; e (iii) a precisão da interação multimodal se dá para determinadas tarefas [Balbo *et al*., 2003]; [Kaster *et al*., 2003].

1.1.5 Abordagens para a Avaliação da Usabilidade de Dispositivos Móveis e Multimodais

No tocante à condução de ensaios de interação, podem ser percebidas algumas

estratégias-chave utilizadas, tanto dentro quanto fora de um ambiente laboratorial, (Quadro 5) na investigação da usabilidade de dispositivos/ sistemas móveis.

Quadro 5 – Principais estratégias para realização de ensaios de interação com dispositivos móveis.

	Ambiente	Estratégias
Ensaio de Interação	**Laboratório**	▸ Uso de tripé de fixação; ▸ Limitação da área de movimentação; ▸ Usuário sentado e avaliador posicionado com câmera de vídeo atrás do usuário;
	Campo	▸ Uso de micro-câmera; ▸ Avaliador segue usuário com câmera de vídeo.

De uma forma mais abrangente, Hagen *et al.* (2005) apresentaram um *framework* das metodologias emergentes na área de interação homem-máquina para sistemas/ dispositivos móveis (Quadro 6).

Quadro 6 – Métodos emergentes para avaliação de dispositivos e sistemas móveis.

Abordagem	Descrição	Técnicas
Coleta de Dados Mediada	**Usuários e equipamentos móveis mediam a coleta de dados sobre o uso em ambientes naturais.**	
Faça	Participantes fazem a coleta de dados.	*Self-reporting*, diários
Use	Participantes fazem uso da tecnologia e os dados são capturados automaticamente.	Ferramentas de *logging.*
	Participantes seguem sua rotina diária, mas fazem uso de dispositivos de registro (e.g. sensores ou câmeras).	Ferramentas de *logging*, registros em vídeo.
Simulações e Ensaios	**Simulações e ensaios são usados para tornar disponíveis informações do contexto real de uso.**	
Simulações	Suporte físico, ergonômico ou ambiental são usados mais frequentemente dentro do laboratório para simular aspectos de uso do mundo real.	Testes laboratoriais, cenários, heurísticas, protótipos, emuladores, simuladores.
Ensaios (*Enactments*)	*Role-playing* tradicional é estendido para situações nas quais a tecnologia é utilizada.	Cenários, encenação (*role-playing*), *storyboarding,* prototipação
Combinações	**Métodos já existentes, e/ou coleta de dados mediada e/ou simulações e decretos são combinados para permitir acesso a dados complementares.**	

Fonte: Adaptado de [Hagen, 2005].

A dimensão multimodal traz à luz questões relevantes para os métodos de avaliação usabilidade. Na verdade, cada elemento envolvido no projeto da interface com o usuário pode ter um impacto grande sobre a sua usabilidade. Por exemplo, os resultados dos estudos empíricos das aplicações multimodais revelaram problemas intrínsecos referentes à avaliação da usabilidade de uma MUI, no que diz respeito a diversas dimensões, tais como o uso e interpretação das modalidades, as preferências individuais do usuário por uma modalidade, o contexto de uso (e.g., avaliação laboratorial e/ou de campo, dispositivos móveis), a escolha dos dispositivos de entrada e saída e as técnicas de interação [Bernhaupt *et al.,* 2007].

Para a avaliação da usabilidade de uma aplicação multimodal, é imprescindível avaliar não somente a interface com o usuário em si, mas considerar o(s) modo(s) de

interação existente(s) e os dispositivos de entrada e saída disponíveis. Assim como o projeto de MUI requer a seleção dos pares adequados (dispositivo, técnica de interação), a avaliação tem que abordar esta questão, apesar de que muitas MUI têm na redundância (se disponibilizada) a possibilidade de permitir aos usuários interagirem com a aplicação de várias maneiras para acionar o mesmo comando ou inserir os mesmos dados. Em tais casos, o par (dispositivo, técnica de interação) selecionado pelo usuário poderá ser diferente de um usuário para outro, bem como para o mesmo usuário entre duas tarefas ou usos sucessivos da aplicação [Bernhaupt *et al.,* 2007].

Constata-se na literatura da área (e.g., [Lai, 2004]; [Baille & Schatz, 2005]; [Reis *et al*., 2008], [Jöst *et al*., 2005]) que a maioria dos estudos da usabilidade das MUI explora algum teste com usuário, cujas atividades dos usuários são observadas e registradas, enquanto os usuários estão executando tarefas pré-definidas. Teste com usuário é a estratégia preferida de avaliação, uma vez que permite a investigação de como os usuários interagem e adaptam as tecnologias multimodais, fornecendo valiosas informações sobre a usabilidade e a experiência do usuário.

Há uma série de estudos que discutem a questão de saber se a avaliação das aplicações móveis multimodais deve ser realizada em contexto laboratorial e/ou de campo (e.g., [Lai, 2004]; [Baille & Schatz, 2005]; [Reis *et al*., 2008]), sugerindo que aplicações móveis multimodal deverão ser avaliadas e estudadas no campo por usuários finais, em contextos reais, e com limitações reais.

Os métodos de inspeção (através do uso de *diretrizes de projeto*, *guias de estilos* e *padrões*) podem ser aplicados nas fases iniciais do processo de desenvolvimento de MUI a partir da análise dos esboços (*mock-ups*) e protótipos. A falta de conhecimento ergonômico disponível pode explicar porque os métodos de inspeção têm sido menos freqüentemente empregados com uma exceção de Bowman *et al*. (2002). Os métodos de inspeção têm sido menos utilizados devido à falta de conhecimento não somente em termos de experiência dos especialistas para a concepção de sistemas multimodais, como também devido à falta de diretrizes para cobrir todas as modalidades e combinações potenciais que podem ser encontradas nas MUI.

Bernhaupt *et al.* (2007) se utilizam da revisão sistemática cognitiva para avaliar uma MUI, adaptando-a as especificidades da interface.

Questionários têm sido empregados extensivamente para obter o retorno qualitativo dos usuários (por exemplo, a satisfação, a usabilidade percebida do sistema, e as preferências do usuário por uma modalidade) [Kaster *et al.*, 2003] e a carga de trabalho cognitiva (especialmente utilizando o método NASA-TLX) [Brewster *et al.*, 1994]; [Kjeldskov & Stage, 2004]; [Trevisan *et al.*, 2006]. Frequentemente, questionários têm sido usados em combinação com as técnicas de testes com usuários, tal como apresentado em [Jöst *et al.*, 2005].

Mais recentemente, a simulação e verificação baseada em modelos de especificações de sistema tem sido utilizada para prever problemas de usabilidade, tais como estados inacessíveis do sistema ou detecção de conflito de eventos necessários para fusão. Paternò & Santos (2006) propuseram a combinação de modelos de tarefa baseados na notação CTT (*Concurrent Task Tree*) com múltiplas fontes de dados (e.g., dados do rastreamento de movimentos oculares, vídeos, registros de voz) para melhor compreensão da interação com o usuário.

Bernhaupt *et al.* (2007) apresentaram uma abordagem que combina a verificação baseada em modelos (baseado na simulação de extração de cenários dos modelos) e métodos empíricos para avaliação da usabilidade, com a utilização de um caso de estudo para demonstrar como testes com usuário e verbalização de procedimentos (*thinking aloud)*, podem ser adaptados para atender as peculiaridades e necessidades para avaliação de MUI.

Um resumo dos aspectos supramencionados, relacionados a trabalhos desenvolvidos no âmbito de MUI, é apresentado no Quadro 7.

Quadro 7 – Resumo dos principais aspectos das avaliações de usabilidade observados na literatura revisada.

Autor(es)	Aspectos Relevantes
Klein *et al.,* 2001	Avaliação da usabilidade de MUI centrada nas ações do usuário, dentro de um contexto uso específico e através de uma metodologia tradicional integrando simuladores.
Schapira & Sharma, 2001	Experimento formal para a avaliação de processos interativos usuário-computador a partir de interfaces de voz e gestos manuais. Com a adoção da Parte 9 do padrão ISO 9241 para a avaliação de dispositivos de apontamento.
Oviatt, 2003	Investigação dos fundamentos da ciência cognitiva na interação multimodal e do papel essencial que a modelagem centrada no usuário tem desempenhado na concepção das MUI e a descrição do papel desempenhado pelas metodologias e métricas de avaliação no desenvolvimento de MUI.
Dybkjær *et al*., 2004a	Verificação da existência de lacunas sobre o conhecimento da interação unimodal, referente à avaliação da usabilidade de SLDSs (*Spoken Language Dialogue Systems*) e constatação da necessidade de concepção de novas métricas de avaliação, em face tanto à contínua evolução (sofisticação) destes sistemas quanto à utilização desses sistemas associada a outras modalidades de interação.
Taib & Ruiz, 2005	Discussão sobre a falta de conformidade dos padrões estabelecidos pela indústria para o desenvolvimento de MUI e o conseqüente impacto na concepção de aplicações envolvendo processos interativos multimodais; Proposição de uma abordagem para a identificação das preferências do usuário final durante a fase de desenvolvimento e avaliação dos custos.
Petridis *et al*., 2006	Avaliação da usabilidade de MUI centrada no usuário, através da comparação de diferentes dispositivos de gestos manuais 2D/3D, como entrada do sistema. Com a condução do processo avaliatório a partir de métricas baseadas na tarefa e relacionadas à facilidade de memorização e ao nível de satisfação do usuário.
Stanciulescu *et al*., 2007	Avaliação de MUI baseado em regras ergonômicas quando da combinação de modalidade de voz e caneta magnética; Como as *guidelines* são limitadas e apenas algumas foram validadas, foi realizada a abstração dos resultados deste estudo em uma base de conhecimento de regras ergonômicas sobre especificações de interface aplicada automaticamente em uma ferramenta (tendo já sido validado para as GUI).
Bernhaupt *et al.,* 2007	Abordagem que combina a verificação baseada em modelos (baseado na simulação de extração de cenários dos modelos) e métodos empíricos para avaliação da usabilidade, com a utilização de um caso de estudo para demonstrar como testes com usuário e verbalização de procedimentos (*thinking aloud)*, podem ser adaptados para atender as peculiaridades e necessidades para avaliação de MUI.

1.2 Avaliação Multidimensional de Usabilidade

As estratégias de avaliação podem ser classificadas como pertencentes a uma de quatro grandes categorias, a saber: (i) *centrada na interação usuário-produto*, a qual engloba métodos de observação, entrevistas, verbalização de procedimentos, discussões em grupo, captura automática, dentre outros; (ii) *centrada na inspeção do produto por*

especialistas, composta pelas avaliações heurísticas, diretrizes de projeto, revisões sistemáticas e inspeções formais, de consistência, de características e de padrões; (iii) *centrada em modelos*, na qual se enquadram os métodos de avaliação cognitivos, lingüísticos, físicos e de interação; e (iv) *híbrida (ou adaptativa)*, caracterizada pela integração de duas ou mais categorias previamente citadas [Queiroz, 2001].

A abordagem multidimensional para a avaliação da usabilidade de interfaces para aplicações móveis e multimodais é fundamentada tanto *na interação usuário-produto* quanto *na inspeção do produto por especialistas*. Conforme visualizado na Figura 3, a abordagem fundamenta-se em três dimensões de avaliação, a saber: (i) *inspeção de conformidade do produto a padrões*, expressa na forma de um diagnóstico do processo interativo por um especialista, com base em recomendações consensuais de um padrão ou um conjunto de padrões internacionais; (ii) *mensuração do desempenho do usuário ao utilizar o produto*, expressa sob a forma de análise do desempenho do usuário durante a avaliação da usabilidade; e (iii) *sondagem da satisfação subjetiva do usuário*, expressa pela visão do usuário sobre o processo interativo homem-computador.

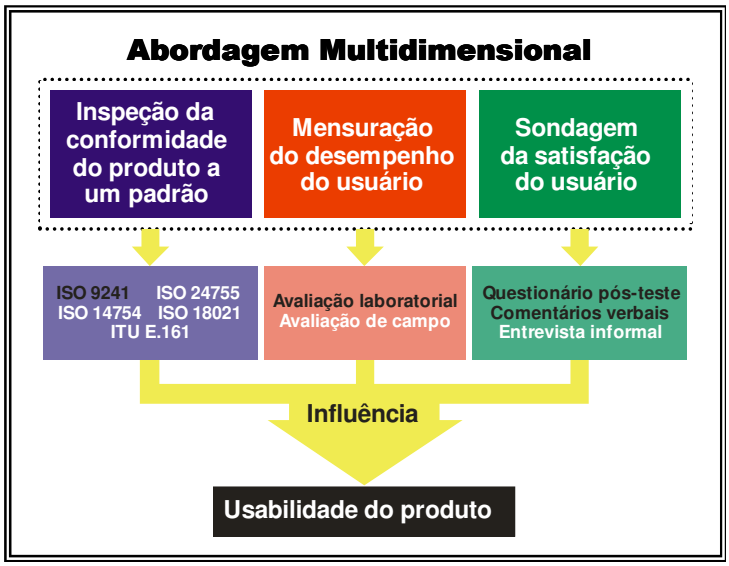


Figura 3 – Descrição da abordagem multidimensional para avaliação da usabilidade de interfaces para aplicações móveis e multimodais.

As dimensões de avaliação que compõem a abordagem multidimensional se complementam, a fim de produzir uma avaliação mais completa do que se aplicadas separadamente, pois possibilita a identificação de falhas de diferentes naturezas.

É importante destacar que a abordagem multidimensional aqui apresentada foi proposta originalmente por Queiroz (2001) e, posteriormente, adaptada aos contextos dos dispositivos móveis por Ferreira (2007) e das interfaces multimodais por Barbosa (2009). As adaptações efetuadas na abordagem original de Queiroz (2001) fundamentaram-se principalmente na escolha: (i) dos padrões, guias de estilo, diretrizes de projeto, etc. a serem adotados; (ii) das técnicas a serem empregadas para a mensuração do desempenho e na definição dos indicadores quantitativos e qualitativos a serem considerados; (iii) dos aspectos a serem levantados a partir dos questionários de sondagem; e (iv) de indicadores objetivos e subjetivos mais significativos para o contexto da avaliação. Cada um destes níveis de escolha depende do contexto, do produto-alvo e das metas e interesses de avaliação [Barbosa, 2009], [Ferreira, 2007].

A abordagem híbrida permite que os enfoques *inspeção de conformidade* e *mensuração do desempenho* detectem problemas de usabilidade de naturezas distintas, enquanto a *sondagem da satisfação do usuário* possibilita corroborar ou refutar os problemas de usabilidade detectados, indicando inclusive o impacto das falhas nas tarefas.

Nas subseções seguintes, são descritos sucintamente a importância e os benefícios consequentes da adoção de cada enfoque.

1.2.1 Inspeção da Conformidade do Produto a Padrões

Nos últimas décadas, a indústria de *software* tem se mostrado mais preocupada com a padronização de algumas partes do *software (e.g. linguagens ou protocolos de comunicação)* como já se fazia com itens de *hardware*.

Várias argumentações têm sido utilizadas para justificar a importância da padronização voltada para a ergonomia de produtos de *software*, dentre as quais: (i) a focalização na consistência tanto em nível da facilidade de aprendizado quanto da facilidade de memorização dos mecanismos de interação pelo usuário; (ii) a otimização das práticas de projeto e avaliação de interfaces, o que pode conduzir à disponibilização de produtos de software mais usáveis e, por conseguinte, melhores do ponto de vista do mercado de consumo; (iii) o aumento do conforto e bem-estar do usuário e, por conseguinte, de sua satisfação; (iv) a priorização apropriada das questões relativas à interface usuário-computador no contexto do desenvolvimento do produto; (v) o cumprimento de requisitos legais, e.g. segurança e saúde no trabalho com dispositivos de visualização; e (vi) o aumento da produtividade do usuário, promovido pela otimização da usabilidade do produto [Ferreira, 2007].

Foram utilizados, no âmbito da abordagem multidimensional, os seguintes padrões:

- ISO 9241 – *Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs)* [ISO, 2001] - tem por objetivo principal auxiliar projetistas no desenvolvimento de sistemas com terminais de visualização de acordo com regras ergonômicas.

- ISO 14754 – *Pen-Based Interfaces – Common gestures for Text Editing with Pen-Based Systems* [ISO, 1999b] - define os comandos gestuais básicos e ações para execução de tais comandos para edição de texto em sistemas baseados em reconhecimento de escrita, e.g. seleção, exclusão, quebra de linha, cópia, colagem, recorte, desfazimento e refazimento de ações;

- ISO 18021 – *User interfaces for mobile tools for management of database communications in a clientserver model* [ISO, 2002] - define funções da interface com o usuário para gerenciar a comunicação com banco de dados do lado cliente de um MBT (*MoBile Tool*) capaz de trocar informações com um servidor;

- ISO 24755 – *Screen icons and symbols for personal mobile communication device* [ISO, 2006b] - define e recomenda um conjunto de ícones e símbolos que, normalmente, estão presentes em dispositivos de comunicação móvel pessoal. Suas recomendações de ícones e símbolos estão relacionadas às características de configuração do próprio dispositivo (e.g. teclado, bateria, toques e sons) e às suas aplicações (e.g. catálogo de endereços, imagens, filmes, áudio, acesso à *Web*, jogos, agenda, mensagens); e

- E.161 – *Arrangement of digits, letters and symbols on telephones and other devices that can be used for gaining access to a telephone network* [ITU, 2001], define recomendações relacionadas ao posicionamento dos dígitos, letras e símbolos em telefones e outros dispositivos móveis.

As tarefas de escritório (*office tasks*) citadas no padrão ISO 9241 incluem uma grande variedade de tarefas de processamento de texto e de dados. Devido à similaridade destas tarefas com as tarefas executadas em outros ambientes, e.g. científico, de telecomunicação, salas de controle e acesso público, os requisitos especificados neste padrão mostram-se apropriados para tais ambientes. Portanto, algumas partes deste padrão podem ser aplicadas para o contexto das tarefas executadas com aplicações móveis e multimodais, e.g., reconhecimento de escrita, reconhecimento de voz, teclado virtual. Uma vez que não havia, no âmbito das organizações internacionais de padronização, um padrão equivalente ao padrão ISO 9241, que envolvesse requisitos ergonômicos para o trabalho com dispositivos móveis e MUI, investigou-se a adequação do padrão ISO 9241 – *Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs)* [ISO, 2001] – a processos de avaliação de dispositivos com aplicações móveis e multimodais e ao escopo delimitado para o produto-alvo. Foram utilizadas, no âmbito da abordagem multidimensional, as partes 11 [ISO, 1998a], 14 [ISO, 1997], 16 [ISO, 1999a] e 17 [ISO, 1998b] do padrão ISO 9241.

O padrão ISO 9241 está dividido em 32 partes. No Quadro 8, são apresentadas algumas partes que compõem atualmente o padrão, dado que novas partes foram criadas e outras já se tornaram obsoletas.

Quadro 8 – ISO 9241 – Essência das partes constituintes.

ISO 9241 *Ergonomic requirements for office work with visual display terminals (VDTs)*	
Partes	**Essência**
1:1997	Considerações introdutórias
2:1992	Orientação sobre requisitos de tarefas
4:1998 9:2000	Considerações ergonômicas relativas ao *hardware*
5:1998 6:1999	Considerações ergonômicas relativas a estações de trabalho e ambientes
11:1998 12:1998 13:1998 14:1997 15:1997 16:1999 17:1998	Considerações ergonômicas relativas ao *software*

Fonte: Adaptado de [Queiroz, 2001].

A Parte 11*, Especificações de usabilidade* (*Guidance on usability)* foi escolhida como referência para a escolha das estratégias de avaliação envolvidas como um todo, a partir da orientação fornecida sobre a avaliação da usabilidade de interfaces em termos de medidas de desempenho e satisfação do usuário. As demais partes, dentre as quais consideram os modos de interação - *menus*, *manipulação direta e formulários*, respectivamente tratados nas *Partes 14*, *16 e 17* do padrão ISO 9241, são direcionadas para o processo de inspeção de conformidade do produto, ou seja, dos modos de interação oferecidos pelo produto avaliado à luz das recomendações da ISO.

É importante destacar que tais padrões, apesar de serem, em geral, adequados às interfaces de aplicações móveis e multimodais, estão intimamente associados ao alvo da avaliação, i.e. a aplicabilidade do padrão está ligada ao produto-alvo da avaliação, assim

como ao aspecto que se deseja avaliar.

1.2.2 Mensuração do Desempenho do Usuário

A mensuração do desempenho consiste em avaliar o uso do produto-alvo por um conjunto representativo de usuários, durante a realização de um conjunto de tarefas típicas. Além da coleta de dados quantitativos, ainda é possível coletar dados de natureza qualitativa através dos comentários verbais feitos pelos próprios usuários de teste durante as sessões.

A mensuração do desempenho dos usuários ao interagir com produtos de *software* é imprescindível por diversas razões, dentre elas: (i) apóia a verificação de produtos, no tocante à adoção de especificações de usabilidade; (ii) auxilia nas tomadas de decisões de projeto; e (iii) possibilita a quantificação das metas da usabilidade (*eficiência*, *eficácia*, *facilidade de aprendizado*, *facilidade de memorização*, *prevenção e recuperação de erros*).

Uma vez que aspectos relacionados às dimensões, à mobilidade e ao contexto de uso dos dispositivos móveis mostra-se significativamente distinto do contexto de uso de dispositivos/sistemas *desktop*, a metodologia prevê a mensuração do desempenho dos usuários típicos tanto em laboratório quanto no campo. Desta forma, aproxima-se o ambiente de teste do ambiente real de uso de aplicações dotadas de MUI em dispositivos móveis. No tocante ao contexto de uso, ainda é possível incluir variações no ambiente de teste de maneira a considerar níveis distintos de ruído (e.g., *silencioso*, *normal* e *ruidoso*).

No tocante aos indicadores quantitativos, são considerados na metodologia: (i) o *tempo de conclusão da tarefa*; (ii) o *número de ações incorretas*; (iii) o *número de escolhas incorretas*; (iv) o *número de erros repetidos;* (v) o *número de consultas à ajuda*; (vi) o *número de ações de reconhecimento dos comandos de voz incorretas*; e (vii) o *número de ações incorretas, erros repetidos e consultas à ajuda associadas às modalidades de entrada de dados (e.g., reconhecimento de escrita).*

Quanto aos indicadores qualitativos são considerados: (i) *facilidade de uso do produto*; (ii) *facilidade de uso dos mecanismos de entrada de dados*; (iii) *facilidade de uso dos modos de entrada de texto*; (iv) *facilidade de compreensão dos termos e símbolos do produto*; (v) *facilidade de compreensão das mensagens de erro/advertência do produto*; (vi) *eficácia da ajuda*; e (vii) *facilidade de uso da aplicação que permite o modo de interação por voz.*

Esta dimensão mostra-se também pertinente, pois avalia o produto sob o ponto de vista do usuário final, expondo suas expectativas e necessidades ao usar o produto.

1.2.3 Sondagem da Satisfação Subjetiva do Usuário

A satisfação subjetiva do usuário também deve ser empregada como fator de avaliação de produtos, pois a satisfação está diretamente ligada ao aumento da produtividade, ao uso agradável e a aceitabilidade do produto pelo usuário.

A estratégia mais empregada nas avaliações para análise das atitudes, opiniões e preferências dos usuários tem sido o uso de questionários como instrumentos para o *delineamento do perfil* e a *sondagem da satisfação subjetiva do usuário*.

No âmbito desta abordagem multidimensional, a aplicação de questionários para o delineamento do perfil dos participantes do ensaio de usabilidade propicia a coleta de

séries de dados qualitativos concernentes a características (i) *físicas*; (ii) *associadas ao conhecimento e à experiência*; e (iii) *associados à tarefa e ao trabalho* do usuário de teste. Por outro lado, a sondagem da opinião dos usuários de teste sobre o produto avaliado produziu séries de dados qualitativos referentes (i) ao *uso e navegação*; (ii) à *documentação online e offline*; e (iii) a *impressões pessoais*.

Indicadores de satisfação também são adquiridos a partir (i) das *respostas verbais coletadas nas entrevistas não estruturadas (debriefing*) realizadas ao final das sessões de teste; (ii) do *índice de satisfação* obtido a partir da administração eletrônica do questionário de sondagem da satisfação do usuário (ferramenta *WebQuest* [Queiroz, 2005]; [Oliveira, 2005]; [WebQuest, 2009]); e (iii) dos *comentários verbais de opiniões feitos pelos participantes durante as sessões de teste*.

1.2.4 A Abordagem Multidimensional e o Padrão ISO 9241-11

Conforme citado na subseção 1.1.1, a Parte 11 do padrão ISO 9241 [ISO, 1998a] define usabilidade como a *eficácia*, *eficiência* e *satisfação* com as quais usuários específicos atingem metas específicas em ambientes específicos, i.e. em um contexto de uso específico. Para determinar o nível de usabilidade alcançado, é necessário medir o desempenho e a satisfação dos usuários durante o uso do produto, mensurando a usabilidade com pelo menos um indicador em cada um dos aspectos (eficácia, eficiência, satisfação).

A abordagem multidimensional atende aos requisitos de usabilidade definidos na Parte 11 do padrão internacional ISO 9241 [ISO, 1998a], uma vez que, principalmente, os enfoques *Mensuração do Desempenho* e *Sondagem da Satisfação Subjetiva do Usuário* são consoantes ao que é definido no padrão. A *Inspeção de Conformidade* também se associa positivamente a esta definição, uma vez que se a inspeção for realizada antes da Mensuração do Desempenho, os resultados advindos do processo de inspeção poderão auxiliar na definição dos objetivos da mensuração do desempenho, focando áreas-alvo a serem observadas quando do uso do produto pelo usuário.

Conforme pode ser visto na Figura 4, no tocante à eficiência, o indicador adotado é o *tempo de execução das tarefas*. No que diz respeito à *eficácia*, os indicadores utilizados serão o *número de ações incorretas*, o *número de opções incorretas*, o *número de erros repetidos*, o *número de consultas à ajuda* e o *número de ações de reconhecimento de voz incorretas*. Como indicadores de satisfação, a metodologia utiliza, além dos *comentários verbais* (feitos pelos usuários durante as sessões de teste), as *respostas verbais coletadas na entrevista não estruturada* e o *índice de satisfação* obtido a partir dos dados coletados pela aplicação do questionário de satisfação.

Dependendo do contexto de uso do produto-alvo e das metas e interesses de avaliação, as adaptações podem ser efetuadas na escolha: (i) do(s) padrão(ões), guia(s) de estilo, diretriz(es) de projeto, etc., a ser(em) utilizado(s); (ii) da(s) técnica(s) a ser(em) adotada(s) nos processos de definição dos indicadores quantitativos e qualitativos e de mensuração do desempenho; e (iii) dos aspectos a serem investigados por meio dos questionários de sondagem do perfil do usuário e da satisfação subjetiva do usuário.

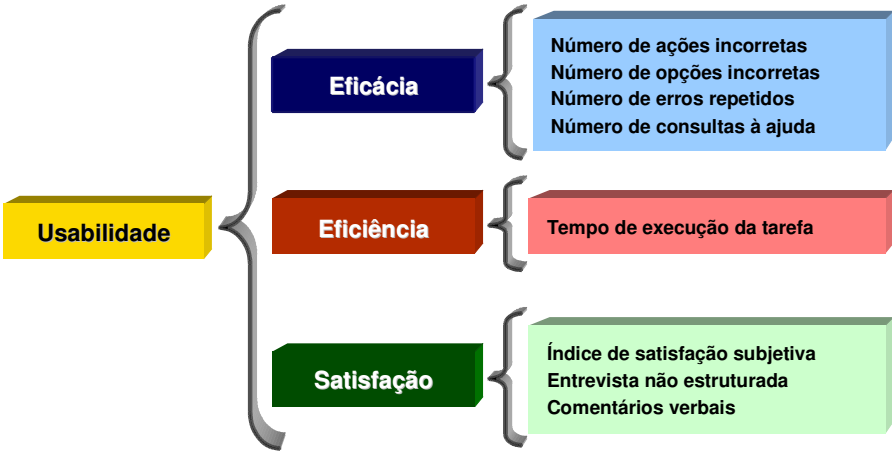


Figura 4 - Indicadores de usabilidade da abordagem híbrida adotada.

1.3 Metodologia de Avaliação

A metodologia é composta por 6 etapas, a saber: (i) *planejamento dos experimentos avaliatórios*; (ii) *treinamento do universo amostral*; (iii) *elaboração do material do ensaio*; (iv) *condução do ensaio e coleta de dados*; (v) *tabulação e análise dos dados*; e (vi) *apresentação dos resultados*. Cada uma destas etapas são apresentadas nas subseções seguintes.

Diversas categorias de produtos já tiveram sua usabilidade avaliada a partir desta metodologia dentre as quais podem ser citadas: (i) aplicações *desktop* (e.g. *MatLab*); (ii) dispositivos móveis (e.g. *Nokia 770 Internet Tablet, HP iPAQ 910c*); e (iii) interfaces multimodais (e.g. *HP Touchsmart PC*). No entanto, no escopo deste mini-curso os exemplos apresentados estão relacionados ao estudo de caso efetuado com o *smartphone* HP iPAQ 910c (Figura 5). Quando necessário, serão apresentados tais exemplos ao final das subseções seguintes.

Figura 5 – HP iPAQ 910c Business Messenger.

Fonte: (HP, 2009) © 2009 Hewlett-Packard Development Company, L.P.

1.3.1. Planejamento dos Experimentos Avaliatórios

A etapa de planejamento dos experimentos avaliatórios é composta por 10 subetapas, conforme pode ser visualizado a seguir. Nesta etapa, encoraja-se também a realização da inspeção de conformidade do produto-alvo a padrões de forma a auxiliar no planejamento dos ensaios com usuários de teste.

Subetapa 1: Definição do produto e do escopo de avaliação

O primeiro passo para dar início ao processo de avaliação da usabilidade consiste na seleção do produto-alvo e na definição do escopo de avaliação, i.e., decidir quais aspectos de *software, hardware, mecanismos de entrada e saída de dados, dentre outros,* serão focalizados.

Subetapa 2: Definição das Metas e Interesses

Consiste na definição dos objetivos gerais e específicos que fundamentarão a condução da avaliação.

Subetapa 3: Seleção de padrões para a Inspeção de Conformidade

Consiste na seleção dos padrões adequados ao produto-alvo, ao escopo da avaliação e aos objetivos gerais e específicos do processo avaliatório.

Subetapa 4: Realização da Inspeção de Conformidade

Embora a inspeção de conformidade possa ser realizada em outro momento, encoraja-se sua realização nesta subetapa da metodologia, pois os resultados advindos de tal inspeção poderão direcionar, de maneira mais eficaz, a condução da mensuração do desempenho e da sondagem da satisfação subjetiva do usuário. A realização da inspeção proporcionará ao avaliador maior conhecimento do produto, permitindo sua focalização em problemas-alvo, além de uma seleção mais adequada de cenários de teste para a subetapa de mensuração do desempenho do usuário e um melhor direcionamento das questões durante a sondagem de sua satisfação subjetiva. Ao final da inspeção de conformidade, conforme recomendado pela ISO, deverá ser calculada a *taxa de adoção* do produto às recomendações contidas no padrão. Foram utilizados os padrões ISO 9241 (partes 14, 16 e 17), ISO 14754 e ISO 24755.

Subetapa 5: Caracterização do Universo Amostral

Consiste na especificação das características relevantes para o delineamento do(s) perfil(s) dos usuários de teste, tais como faixa etária, grau de escolaridade, nível de conhecimento em informática, nível de conhecimento prévio do produto, familiaridade com outros idiomas, dentre outros.

Subetapa 6: Levantamento dos Usuários de Teste Potenciais

Consiste no mapeamento do contingente de potenciais usuários de teste e na sondagem informal da disponibilidade e interesse de participação nos ensaios de testes, levando-se em consideração as características do universo amostral que foram pré-definidas.

Subetapa 7: Definição do Modo de Recrutamento dos Usuários

Compreende a definição e estruturação de uma estratégia de recrutamento dos participantes, e.g. envio de correspondências, aplicação de questionário pré-teste, contato telefônico e/ou email, mediante a qual a sondagem da disponibilidade de interesse dos usuários potenciais dos produtos-alvo será formalizada.

Subetapa 8: Definição do Número de Participantes do Ensaio de Avaliação

Decisão do número de participantes com base nos resultados da sondagem do universo

de potenciais usuários de teste. Nesta subetapa, é necessário dividir a amostra de usuários participantes em categorias, conforme a natureza da avaliação, e definir os ambientes de teste (e.g., laboratório e campo, ambiente silencioso e ruidoso), caso se queira verificar a influência do ambiente no uso do produto avaliado (vide Quadro 15).

Subetapa 9: Seleção das Técnicas de Avaliação

Consiste na definição das técnicas (veja Quadro 1) que serão empregadas para avaliar os produtos-alvo a partir da Mensuração do Desempenho e da Sondagem da Satisfação, fundamentadas tanto em função dos recursos humanos, físicos, orçamentários, materiais e de prazo disponíveis, quanto das informações coletadas nas subetapas anteriores.

Subetapa 10: Definição dos Indicadores Objetivos e Subjetivos

Consiste na escolha dos indicadores objetivos e subjetivos mais significativos para o contexto do processo de avaliação, em função das técnicas de avaliação e dos objetivos geral e específicos previamente selecionados.

Exemplo 01: Planejamento Geral do Processo de Avaliação

Os quadros apresentados a seguir apresentam as decisões tomadas em cada uma das subetapas supradescritas. Nos Quadros 9 a 12, apresenta-se um sumário dos aspectos relacionados ao planejamento geral da avaliação.

O Quadro 9 contém informações que caracterizam o produto e o escopo do processo de avaliação.

Quadro 9 – Escopo da Avaliação.

PRODUTO E ESCOPO DA AVALIAÇÃO		
Produto	***HP iPAQ 910c (smartphone)***	
Escopo de Avaliação	***Hardware***	i. tamanho e peso do dispositivo; ii. localização e facilidade de uso dos botões; iii. localização e facilidade de uso da caneta *stylus*; iv. brilho/reflexo da tela de visualização; v. capacidade e velocidade de processamento.
	Software	i. tela inicial do dispositivo (tela Today); ii. aplicativo Internet Explorer Mobile; iii. aplicativo Calendar; iv. aplicativo HP Voice Commander; v. aplicativo de correio eletrônico; vi. editor de texto (Office Mobile Word); vii. chamada telefônica; viii. aplicativo de reprodução de mídia de áudio (Windows Media Player).
	Modalidades de Interação	i. reconhecimento de voz, ii. reconhecimento de escrita, iii. teclado virtual, iv. manipulação direta via tela sensível ao toque; e v. teclado

No Quadro 10, enunciam-se os objetivos geral e específicos do processo de avaliação, enquanto o Quadro 11 contém um sumário das técnicas de avaliação adotadas em cada

indicadores de usabilidade objetivos e subjetivos considerados no experimento.

Quadro 10 – Objetivos Geral e Específicos da Avaliação.

OBJETIVOS DA AVALIAÇÃO	
Objetivo Geral	Diagnóstico objetivo e subjetivo do processo interativo homem-produto.
Objetivos Específicos	i. Observação *da facilidade de uso do produto*; ii. Observação da *facilidade de execução das tarefas*; iii. Observação da *facilidade de uso dos mecanismos de entrada de dados*; iv. Observação da *eficiência dos modos de entrada de texto*; v. Observação da *clareza dos termos e símbolos* utilizados no produto; vi. Observação da *facilidade de uso da aplicação que permite o modo de interação por voz*; vii. Mensuração do *tempo de conclusão* das tarefas; viii. Mensuração do *número de opções incorretas* durante a execução da tarefa; ix. Mensuração do *número de ações incorretas* durante a execução da tarefa; x. Mensuração do *número de erros repetidos* durante a execução das tarefas; xi. Mensuração do *número de consultas* à ajuda durante a execução das tarefas; xii. Número de *ações de reconhecimento de voz incorretas*.

Quadro 11 – Técnicas de Avaliação Selecionadas.

TÉCNICAS DE AVALIAÇÃO & INDICADORES		
Técnicas de Avaliação	**Inspeção de Conformidade**	Padrões selecionados: i. ISO 9241 – Partes 14, 16 e 17; ii. ISO 14754; iii. ISO 24755;
	Mensuração do Desempenho	i. Observação direta com registro em vídeo (ambiente laboratorial e de campo) ii. Verbalização das ações (*thinking aloud*).
	Sondagem da Satisfação Subjetiva	i. Questionários; ii. Anotações de comentários verbais; iii. Entrevista não estruturada. Observação: Para apoiar o processo de coleta de dados, por meio de questionários, foi utilizada a ferramenta *WebQuest* [Queiroz *et al*., 2005]; [Oliveira, 2005]; [WebQuest, 2009].
Indicadores Objetivos		i. Tempo de execução das tarefas; ii. Número de opções incorretas; iii. Número de ações incorretas; iv. Número de erros – *Writing Pad*; v. Números de erros repetidos – *Writing Pad*; vi. Número de consultas à ajuda – *Writing Pad*; vii. Número de erros – *On-Screen Keyboard*; viii. Números de erros repetidos – *On-Screen Keyboard*; ix. Número de consultas à ajuda – *On-Screen Keyboard*; x. Número de erros – *Character Pad*; xi. Números de erros repetidos – *Character Pad*; xii. Número de consultas à ajuda – *Character Pad*; xiii. Número de erros repetidos; xiv. Número de consultas à ajuda.
Indicadores Subjetivos		i. Facilidade de uso do produto; ii. Facilidade de uso dos mecanismos de entrada de dados; iii. Facilidade de uso dos modos de entrada de texto; iv. Facilidade de compreensão dos termos e símbolos do produto; v. Facilidade de compreensão das mensagens de erro/advertência do produto; vi. Eficiência da ajuda.

Por sua vez, os Quadros 12 a 14 contêm aspectos relacionados ao planejamento do universo amostral de usuários participantes do processo de avaliação.

Quadro 12 – Planejamento do Universo Amostral.

Caracterização do Universo Amostral	
Universo Amostral	Usuários principiantes e intermediários
Usuários de Teste Potenciais	90 usuários
Modo de Recrutamento dos Usuários	Contato pessoal por telefone e/ou e-mail.
Número de Participantes	74 usuários (72 + 2 usuários para sessões piloto)

Quadro 13 – Características dos perfis dos usuários de teste.

Categorias / Caracteristicas	Principiantes	Intermediários
Conhecimento em informática	Básico/Intermediário	Intermediário/Avançado
Experiência com produtos similares	Não	Sim
Familiaridade com a língua inglesa	Sim	Sim

Quadro 14 – Distribuição das amostras de usuários para o *HP iPAQ PC*.

Ambiente	Categoria	Quantidade	
Laboratório	Principiantes	18	36
	Intermediários	18	
Campo	Principiantes	18	36
	Intermediários	18	

1.3.2 Treinamento do Universo Amostral

A segunda etapa da metodologia envolve a capacitação do universo amostral de teste.

Subetapa 11: Familiarização dos usuários de teste com o produto-alvo

Consiste em fornecer aos usuários de teste selecionados para os ensaios um nível mínimo de familiarização com o produto-alvo a fim de evitar comportamentos divergentes entre os usuários. Dependendo do produto-alvo e do universo amostral esta subetapa pode ser suprimida da abordagem metodológica, i.e. quando não houver necessidade da realização de treinamento. O treinamento dos usuários pode ocorrer de forma individual ou em grupo. Além disso, pode ocorrer neste momento ou, posteriormente, no início da sessão de teste, imediatamente após a introdução do participante no ambiente de avaliação. Neste último caso, recomenda-se o envio antecipado de material instrucional sobre o produto por meio eletrônico.

1.3.3 Elaboração do Material do Ensaio

Esta etapa compreende a elaboração e validação de todo o material que será utilizado na condução dos ensaios com os usuários.

Subetapa 12: Planejamento e Estruturação das Tarefas de Teste

Consiste na elaboração das tarefas que serão relevantes ao contexto de uso do produto-alvo da avaliação, tendo-se principalmente como base o escopo de avaliação do produto, os objetivos geral e específicos da avaliação e os indicadores objetivos e subjetivos predefinidos.

Subetapa 13: Elaboração de Documentos

Consiste na elaboração de 3 documentos principais, a saber: (i) *ficha cadastral do participante* (Figura 6a), cujo propósito é coletar informações pessoais do usuário; (ii) *documento de aceitação das condições de teste* (Figura 6b), que trata das questões éticas relacionadas ao registro em vídeo da sessão de teste; e (iii) *termo de confidencialidade* (Figura 6c), que está relacionado à divulgação de informações relacionadas ao produto.

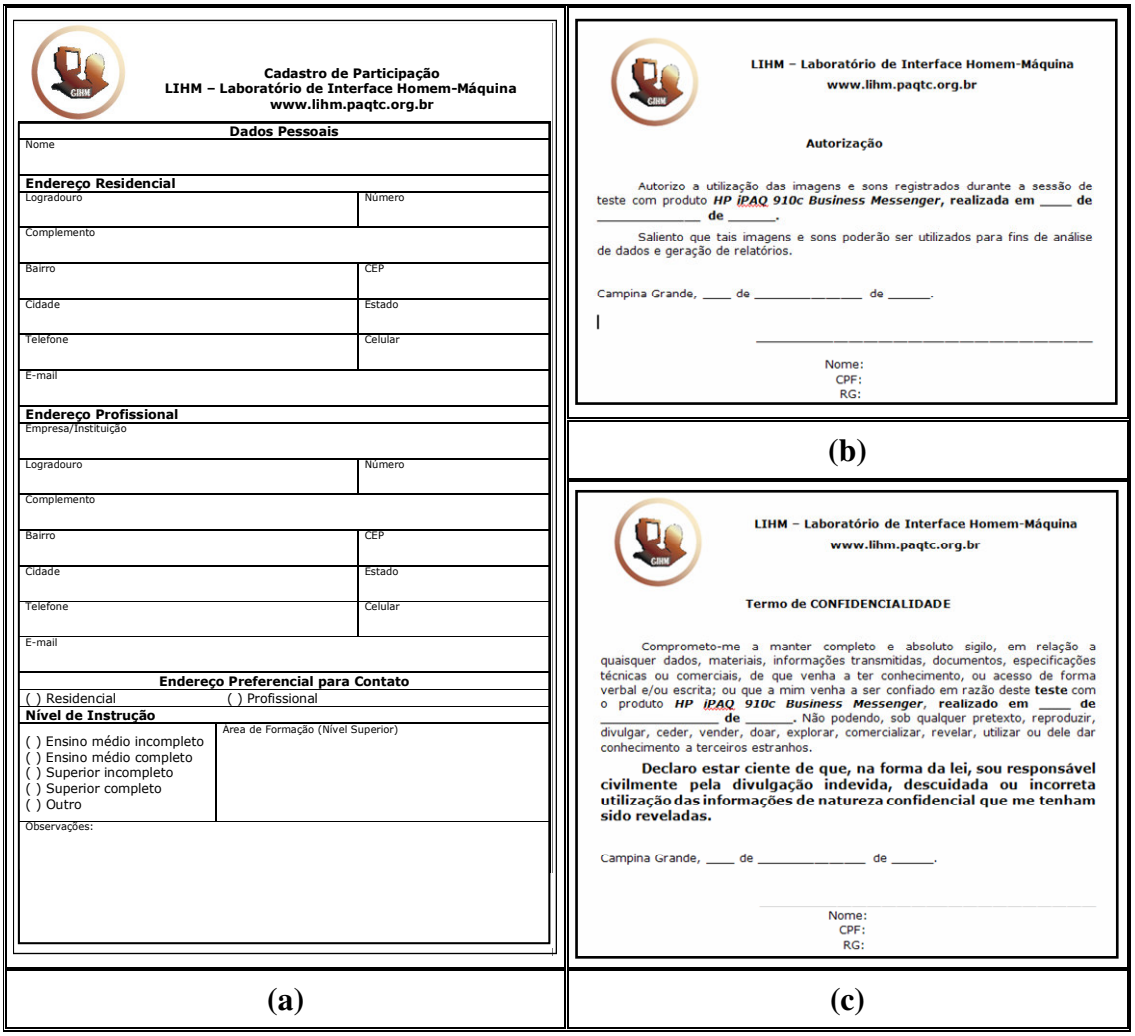


Figura 6 – Documentos principais: (a) Ficha cadastral do participante; (b) Documento de aceitação das condições de teste; e (c) Termo de confidencialidade.

Subetapa 14: Elaboração do Material Necessário à Condução do Processo de Avaliação

Compreende a elaboração dos seguintes materiais: (i) *questionários pré e pós-teste*; (ii) *roteiros das tarefas de teste* (para o usuário e para o avaliador); (iii) *fichas de registro de eventos*; e (iv) *guia para a entrevista não-estruturada*.

O questionário *pré-teste*, usualmente destinado ao delineamento do perfil do usuário, envolve tipicamente questões relacionadas às características físicas (e.g., faixa etária, sexo, grau de instrução, acuidade visual) e às habilidades e conhecimentos dos usuários (e.g., nível de conhecimento em informática e do produto, habilidades de

idioma).

O questionário *pós-teste*, usualmente destinado à sondagem da satisfação subjetiva do usuário, baseou-se, na metodologia ora descrita, em duas premissas: (i) questionamentos alinhados ao escopo e às tarefas pré-definidas, de maneira a confrontar tais dimensões; e (ii) limitação no número de questões, a fim de não causar cansaço ou desconforto ao usuário.

É importante destacar que o *Roteiro das Tarefas de Teste* deve conter tarefas em um nível mais alto, evitando, por exemplo, fornecer indicações de acionamento de teclas, botões ou opções de menu. Recomenda-se a elaboração de duas versões de tal roteiro, uma para o usuário e outra para o avaliador. A versão do avaliador diferencia-se daquela destinada ao usuário por conter indicações de quais objetivos específicos e/ou indicadores objetivos e subjetivos estão sendo avaliados em cada tarefa.

A *Ficha de Registro de Eventos* auxilia o avaliador na documentação dos eventos importantes associados aos indicadores objetivos e subjetivos ocorridos durante a sessão de teste.

O *Guia para a Entrevista Não Estruturada* é composto por tópicos ou questionamentos específicos a serem abordados após a execução das tarefas de teste pelo usuário. Por se tratar de uma entrevista não estruturada e, por conseguinte, informal, o avaliador tem a liberdade de fazer adaptações, de acordo com o comportamento do usuário e o andamento da sessão de teste.

Subetapa 15: Validação do Material Elaborado

Consiste na condução de teste(s)-piloto com objetivo de detectar problemas nos métodos planejados, no material de teste elaborado, no produto e em sua documentação. Esta subetapa compreende também as atividades associadas aos ajustes que devem ser efetuados nos materiais do ensaio e talvez no próprio planejamento dos ensaios. A execução do teste piloto deve seguir as subetapas apresentadas na subseção 1.3.4 (condução do ensaio e coleta de dados).

Exemplo 02: Tarefas de Teste e Artefatos Utilizados na Aquisição de Dados

As tarefas de teste para o *HP iPAQ 910c* foram planejadas de forma a contemplar desde a inicialização do dispositivo até o uso dos seus principais aplicativos e funções, assim como os diversos modos de interação disponibilizados (e.g., tela sensível ao toque, teclado, reconhecimento de voz, reconhecimento de escrita). Além disto, foram considerados os resultados preliminares obtidos a partir da Inspeção de Conformidade do produto aos padrões selecionados. Assim sendo, foram planejadas, inicialmente, 7 tarefas de teste.

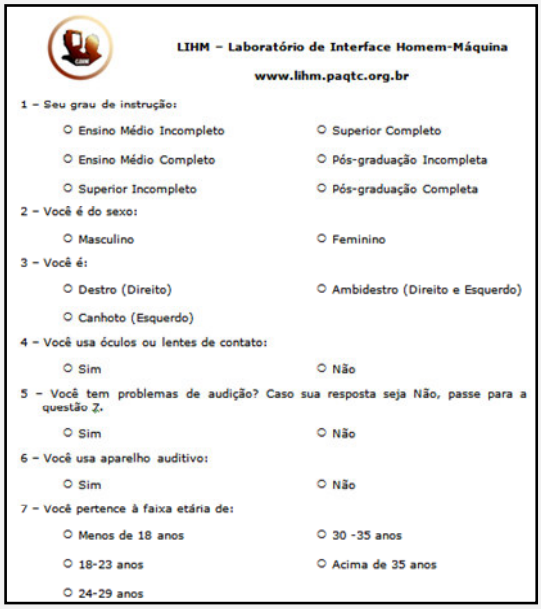
A condução dos testes-piloto, dentre outros aspectos considerados, possibilitou constatar que as tarefas de teste foram subdimensionadas, ultrapassando 60 minutos e tornando-se contraproducentes do ponto de vista do usuário. Assim sendo, as tarefas foram reestruturadas, sendo reduzidas para 6. No Quadro 15, apresenta-se uma síntese das tarefas de teste em sua versão final.

Quadro 15 – Síntese do planejamento das tarefas de teste.

Planejamento das Tarefas de Teste			
			Número de Tarefas de Teste
Natureza do Ensaio	**Laboratorial**		*06*
	De Campo		*06*
Especificação das Tarefas	**Laboratorial (36 usuários)**	**Tarefa 00**	Inicialização do dispositivo
		Tarefa 01	Consulta e agendamento de compromissos
		Tarefa 02	Entrada de dados textuais
		Tarefa 03	Uso do aplicativo de mensagens (*e-mail*)
		Tarefa 04	Chamada telefônica
		Tarefa 05	Uso do aplicativo de reprodução de áudio
	De Campo (36 usuários)	**Tarefa 00**	Inicialização do dispositivo
		Tarefa 01	Consulta e agendamento de compromissos
		Tarefa 02	Entrada de dados textuais
		Tarefa 03	Uso do aplicativo de mensagens (*e-mail*)
		Tarefa 04	Chamada telefônica
		Tarefa 05	Uso do aplicativo de reprodução de áudio

Foram necessários à condução do processo de avaliação: (i) um *Questionário Pré-teste* (Figuras 7a); (ii) um *Questionário Pós-teste* (Figura 7b); (iii) um *Roteiro de Testes* (Figura 8a); (iv) uma *Ficha de Registro de Eventos* (Figura 8b); e (v) um *Guia para a Entrevista Não Estruturada* (Figura 9).

A subetapa de Validação do Material Elaborado se deu a partir da condução de duas sessões de teste-piloto, uma no laboratório e outra em campo, as quais resultaram, conforme supramencionado, em ajustes na quantidade e na reestruturação das tarefas de teste.



LIHM – Laboratório de Interface Homem-Máquina

www.lihm.paqtc.org.br

1 – Seu grau de instrução:

○ Ensino Médio Incompleto ○ Superior Completo

○ Ensino Médio Completo ○ Pós-graduação Incompleta

○ Superior Incompleto ○ Pós-graduação Completa

2 – Você é do sexo:

○ Masculino ○ Feminino

3 – Você é:

○ Destro (Direito) ○ Ambidestro (Direito e Esquerdo)

○ Canhoto (Esquerdo)

4 – Você usa óculos ou lentes de contato:

○ Sim ○ Não

5 – Você tem problemas de audição? Caso sua resposta seja Não, passe para a questão 7.

○ Sim ○ Não

6 – Você usa aparelho auditivo:

○ Sim ○ Não

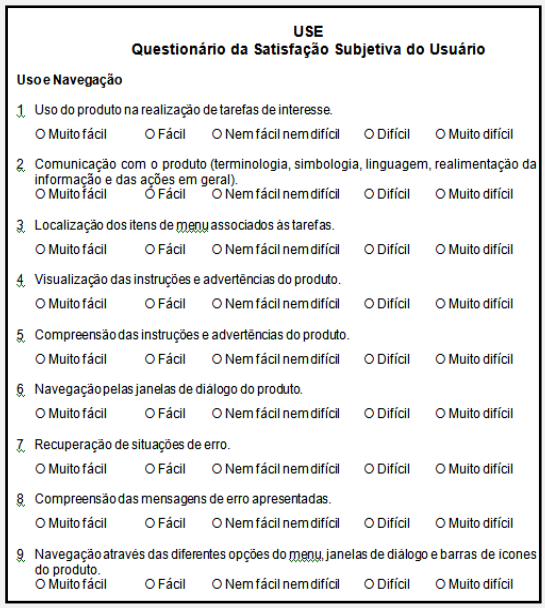
7 – Você pertence à faixa etária de:

○ Menos de 18 anos ○ 30 -35 anos

○ 18-23 anos ○ Acima de 35 anos

○ 24-29 anos

(a)



USE

Questionário da Satisfação Subjetiva do Usuário

Uso e Navegação

1 Uso do produto na realização de tarefas de interesse.
○ Muito fácil ○ Fácil ○ Nem fácil nem difícil ○ Difícil ○ Muito difícil

2 Comunicação com o produto (terminologia, simbologia, linguagem, realimentação da informação e das ações em geral).
○ Muito fácil ○ Fácil ○ Nem fácil nem difícil ○ Difícil ○ Muito difícil

3 Localização dos itens de menu associados às tarefas.
○ Muito fácil ○ Fácil ○ Nem fácil nem difícil ○ Difícil ○ Muito difícil

4 Visualização das instruções e advertências do produto.
○ Muito fácil ○ Fácil ○ Nem fácil nem difícil ○ Difícil ○ Muito difícil

5 Compreensão das instruções e advertências do produto.
○ Muito fácil ○ Fácil ○ Nem fácil nem difícil ○ Difícil ○ Muito difícil

6 Navegação pelas janelas de diálogo do produto.
○ Muito fácil ○ Fácil ○ Nem fácil nem difícil ○ Difícil ○ Muito difícil

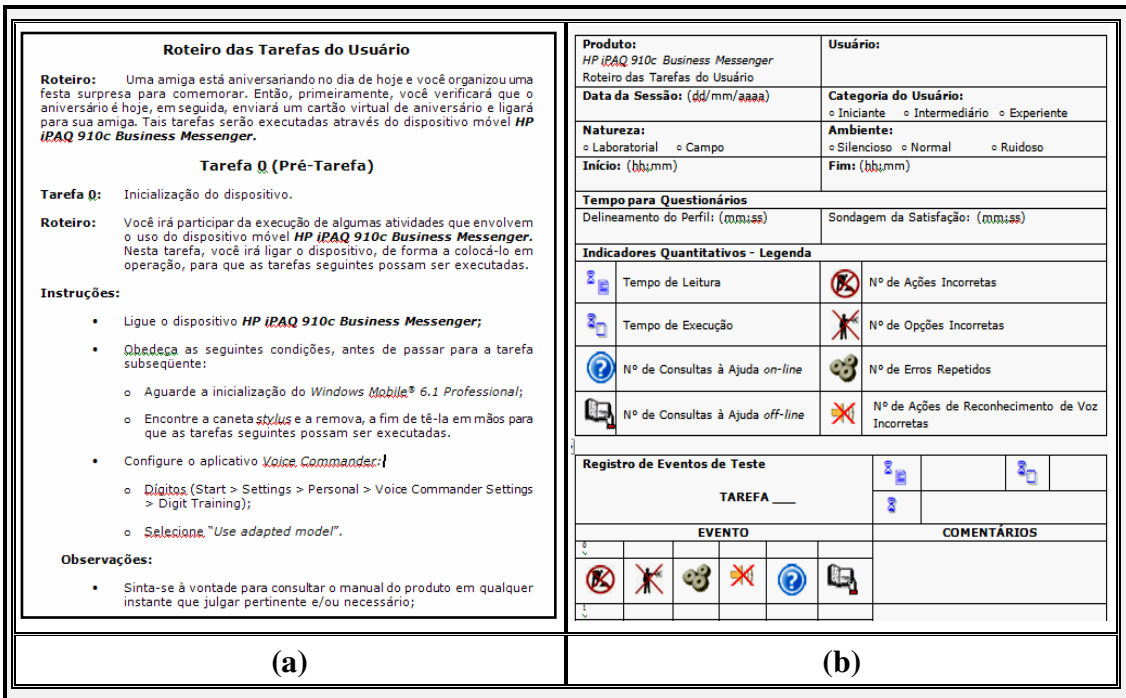
7 Recuperação de situações de erro.
○ Muito fácil ○ Fácil ○ Nem fácil nem difícil ○ Difícil ○ Muito difícil

8 Compreensão das mensagens de erro apresentadas.
○ Muito fácil ○ Fácil ○ Nem fácil nem difícil ○ Difícil ○ Muito difícil

9 Navegação através das diferentes opções do menu, janelas de diálogo e barras de ícones do produto.
○ Muito fácil ○ Fácil ○ Nem fácil nem difícil ○ Difícil ○ Muito difícil

(b)

Figura 7 – Excertos dos questionários: (a) *pré-teste*; e (b) *pós-teste*.



Roteiro das Tarefas do Usuário

Roteiro: Uma amiga está aniversariando no dia de hoje e você organizou uma festa surpresa para comemorar. Então, primeiramente, você verificará que o aniversário é hoje, em seguida, enviará um cartão virtual de aniversário e ligará para sua amiga. Tais tarefas serão executadas através do dispositivo móvel ***HP iPAQ 910c Business Messenger.***

Tarefa 0 (Pré-Tarefa)

Tarefa 0: Inicialização do dispositivo.

Roteiro: Você irá participar da execução de algumas atividades que envolvem o uso do dispositivo móvel ***HP iPAQ 910c Business Messenger.*** Nesta tarefa, você irá ligar o dispositivo, de forma a colocá-lo em operação, para que as tarefas seguintes possam ser executadas.

Instruções:

- Ligue o dispositivo ***HP iPAQ 910c Business Messenger;***
- Obedeça as seguintes condições, antes de passar para a tarefa subseqüente:
  - Aguarde a inicialização do *Windows Mobile® 6.1 Professional*;
  - Encontre a caneta *stylus* e a remova, a fim de tê-la em mãos para que as tarefas seguintes possam ser executadas.
- Configure o aplicativo *Voice Commander:*
  - Dígitos (Start > Settings > Personal > Voice Commander Settings > Digit Training);
  - Selecione *"Use adapted model"*.

Observações:

- Sinta-se à vontade para consultar o manual do produto em qualquer instante que julgar pertinente e/ou necessário;

Produto: *HP iPAQ 910c Business Messenger* Roteiro das Tarefas do Usuário	**Usuário:**
Data da Sessão: (dd/mm/aaaa)	**Categoria do Usuário:** ○ Iniciante ○ Intermediário ○ Experiente
Natureza: ○ Laboratorial ○ Campo	**Ambiente:** ○ Silencioso ○ Normal ○ Ruidoso
Início: (hh:mm)	**Fim:** (hh:mm)
Tempo para Questionários	
Delineamento do Perfil: (mm:ss)	Sondagem da Satisfação: (mm:ss)

Indicadores Quantitativos - Legenda			
	Tempo de Leitura		Nº de Ações Incorretas
	Tempo de Execução		Nº de Opções Incorretas
	Nº de Consultas à Ajuda *on-line*		Nº de Erros Repetidos
	Nº de Consultas à Ajuda *off-line*		Nº de Ações de Reconhecimento de Voz Incorretas

Registro de Eventos de Teste TAREFA ___								
EVENTO							**COMENTÁRIOS**	

(a) **(b)**

Figura 8 – Excertos: (a) do *Roteiro de Testes*; (b) da *Ficha de Registro de Eventos*.

Entrevista Informal ou Coloquial

Sobre o produto:

1 – Uso da Ajuda *on-line* e off-*line*.

2 – O que você achou do modo de interação?

Uso dos botões; Caneta *stylus* e Comandos de voz

3- O que você achou dos modos de entradas de dados textuais?

Uso do teclado virtual

Uso do teclado

Uso do reconhecimento de escrita (*Letter Recognizer;Block Recognize; Transcriber)*

4 – O que você achou da apresentação da interface? Menus, ícones, cores...

5 – O que você achou do *layout* do dispositivo? (Tamanho, peso, botões, caneta *stylus*)

6 - O que você achou do acesso à Web através deste dispositivo?

7 – O que você achou da leitura de informações através deste dispositivo?

8 – De forma geral, gostou ou não do produto?

Sobre a sessão de teste:

1 – O ambiente é agradável? Você acha que, de alguma forma influenciou na execução das tarefas?

2 – O equipamento de gravação lhe incomodou? Influenciou na sua forma de interação?

3 – O que você achou das tarefas de teste?

Figura 9 – Guia para a entrevista não estruturada.

1.3.4. Condução do Ensaio e Coleta de Dados

Esta subetapa da metodologia caracteriza-se pela: (i) aplicação das técnicas pré-

definidas de avaliação da usabilidade; e (ii) coleta de indicadores objetivos e subjetivos.

Uma sessão típica de teste compreende as subetapas descritas a seguir.

Subetapa 16: Introdução do Participante no Ambiente de Teste

Consiste no esclarecimento de tópicos sobre usabilidade, do propósito do teste e dos procedimentos a serem seguidos durante a sessão. Além da abordagem das questões éticas envolvidas. Para o avaliador, destaca-se o comprometimento com o sigilo e anonimato dos usuários, quanto aos seus dados pessoais e ao relatório de falhas. Para os usuários, o compromisso em não divulgar informações do produto testado, por meio da assinatura do *Termo de Confidencialidade*. Na ocasião, os usuários devem optar (ou não) pela autorização do uso do áudio e vídeo de sua sessão para elaboração de relatórios multimídia, a partir do *Documento de Aceitação das Condições de Teste*.

Subetapa 17: Realização de Treinamento para o Fornecimento de mais Informações sobre o Produto

O avaliador descreve as principais funções e formas de uso do produto ao usuário. Nesta ocasião, são esclarecidas as potenciais dúvidas do usuário, relacionadas a cada produto, nas suas respectivas sessões de teste. Caso já tenha ocorrido um treinamento prévio ou não seja necessário, dado o contexto do produto e da categoria de usuários participantes, esta subetapa pode ser suprimida da abordagem.

Subetapa 18: Aplicação de Questionário para Delineamento do Perfil do Usuário (Pré-Teste)

Consiste na administração de um questionário para coletar dados sobre o perfil dos usuários (e.g. sexo, faixa etária, grau de instrução, experiência computacional prévia, familiaridade com a língua inglesa) e dados relacionados ao conhecimento prévio do usuário com o produto avaliado. O questionário pode ser aplicado em formato impresso ou eletrônico, a partir de uma ferramenta de administração de questionários (e.g., [WebQuest, 2009]).

Subetapa 19: Execução do Roteiro das Atividades de Teste

Descrição: Corresponde à execução das tarefas de teste pelos usuários a partir do uso do roteiro criado para avaliação, com o registro em vídeo e coleta de indicadores objetivos pré-definidos e comentários relativos aos indicadores subjetivos (por meio da Ficha de Registro de Eventos). O avaliador exerce o papel de observador, interrompendo minimamente na execução das tarefas pelo usuário.

Subetapa 20: Aplicação de Questionário para Sondagem da Satisfação Subjetiva do Usuário (Pós-Teste)

Consiste na coleta de dados relativos à satisfação do usuário ao utilizar o produto, à navegação pela interface, à interação pelos diferentes modos de comunicação e à documentação. O questionário pode ser aplicado em formato impresso ou eletrônico por meio de uma ferramenta de administração de questionários (e.g., [WebQuest, 2009]).

Subetapa 21: Realização de Entrevista Não Estruturada

Consiste na condução de uma entrevista informal, a fim de coletar mais informações

sobre as impressões do usuário e esclarecer comportamentos e/ou comentários ocorridos durante a sessão de teste.

Exemplo 03: Descrição do(s) Ambiente(s) de Teste e do Material Utilizado

No estudo de caso ora apresentado, optou-se pela condução do ensaio em dois contextos de uso, no laboratório e no campo. A Figura 10 contém uma vista de topo do ambiente laboratorial utilizado – o *Laboratório de Interfaces Homem-Máquina* (LIHM) do Centro de Engenharia Elétrica e Informática (CEEI) da Universidade Federal de Campina Grande (UFCG). O ambiente de campo compreendeu a infra-estrutura predial do Departamento de Sistemas e Computação (DSC) da UFCG.

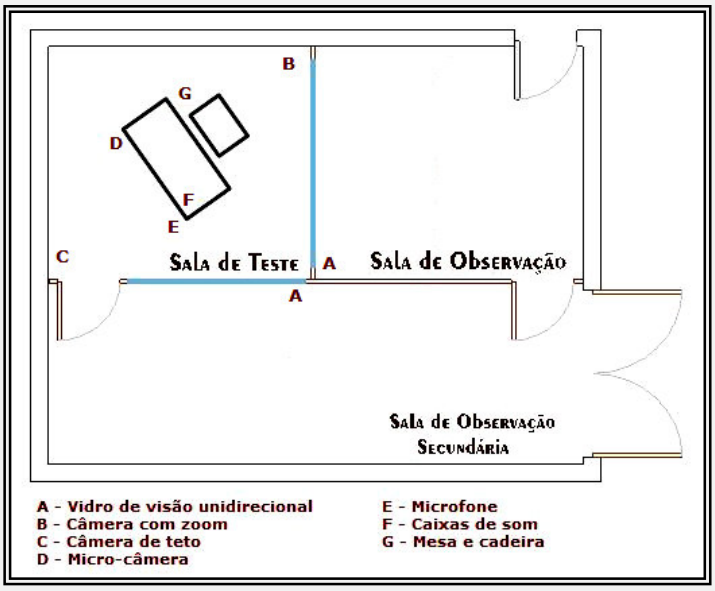


Figura 10 – Vista de topo do *LIHM/CEEI/UFCG.*

No Quadro 16, lista-se o material utilizado em ambos os ambientes de avaliação.

Quadro 16 – Síntese do material utilizado nas sessões de teste.

Material		
	Laboratório	**Campo**
Hardware	HP iPAQ 910c Business Messenger; Computadores (2); Câmeras de vídeo (2); Microfones (2).	HP iPAQ 910c Business Messenger; Computadores (2); Mico-câmera de vídeo sem fio (1); Aparato para fixar micro-câmera ao produto; (1); Adaptador de vídeo USB 2.0 (1).
Software	VNC (*Virtual Network Computing*) - software para captura da tela do produto; CamStudio – software para registro em vídeo da tela do produto; WebQuest – ferramenta de automação de questionários pré-teste e pós-teste.	
Outros	Manual do HP iPAQ 910c Business Messenger; Cronômetro (1); CDs/DVDs para backup dos vídeos; Ficha Cadastral do Participante; Documento de Aceitação das Condições de Teste; Termo de Confidencialidade; Roteiro das Tarefas de Teste (versão do usuário e do avaliador); Ficha de Registro de Eventos; Guia para Entrevista Não Estruturada.	

O registro em vídeo das sessões de teste realizadas em campo foi auxiliado por um segundo equipamento de suporte, destinado a acoplar a micro-câmera sem fio ao dispositivo avaliado (Figura 11).

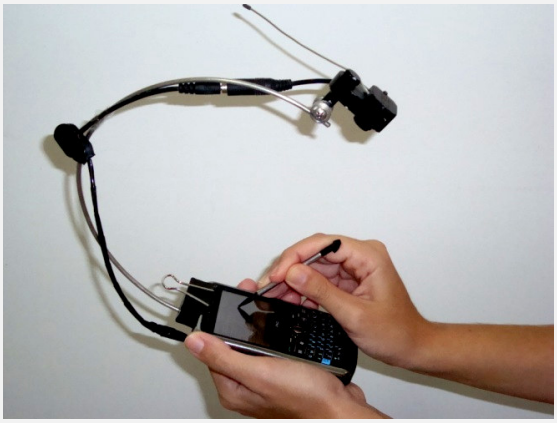


Figura 11 – Aparato para acoplamento da micro-câmera ao dispositivo testado durante a realização dos testes de campo.

1.3.5. Triagem, Análise e Triangulação dos Dados

Esta subetapa consistiu nas atividades associadas à análise, triagem e síntese dos dados coletados na subetapa anterior, tendo sido dividida em quatro subetapas, a saber:

Subetapa 22: Triagem Preliminar dos Dados

Execução de uma análise preliminar dos dados coletados, a fim de detectar problemas colaterais não previamente evidenciados (e.g., se for o caso, desconsiderar os dados coletados em uma sessão de teste atípica).

Subetapa 23: Tabulação e Síntese dos Dados

Consiste na tabulação dos dados obtidos na inspeção de conformidade, durante a mensuração do desempenho do usuário e na sondagem da satisfação subjetiva dos participantes do teste. Recomenda-se a sumarização em tabelas dos seguintes dados: (i) taxas de adoção do produto aos padrões; (ii) indicador de satisfação subjetiva dos usuários; e (iii) síntese da sondagem do perfil dos usuários.

Nesta subetapa, caso seja pertinente, recomenda-se a realização de análises estatísticas nos dados quantitativos sumarizados, como por exemplo, valores mínimo e máximo, média, desvio padrão, variância. Além disto, pode-se optar pelo uso do *teste F ANOVA fator único* [Levine *et al.,* 2000] para verificar a existência de diferenças entre as médias de grupos de dados. Uma vez que este teste evidencia apenas diferenças entre médias, não possibilitando comparações entre pares de grupos, é necessário utilizar adicionalmente, para este propósito, o *procedimento Tukey-Kramer* [Levine *et al.,* 2000]. Por outro lado, para verificar a existência de diferenças estatisticamente significativas entre as médias de dois grupos considerados, pode-se utilizar o teste *t de Student*.

Subetapa 24: Organização dos Problemas Listados

Compreende a organização dos problemas de usabilidade evidenciados a partir da condução dos processos de avaliação, a saber: (i) falhas detectadas na inspeção de conformidade; (ii) falhas detectadas durante a mensuração do desempenho dos usuários; (iii) falhas detectadas no processo de sondagem da satisfação subjetiva dos usuários. Recomenda-se ainda a categorização das falhas encontradas e a elaboração de pareceres parciais do produto sob a perspectiva de cada dimensão de avaliação. Uma sugestão de categorização é apresentada no Quadro 17.

Quadro 17 – Síntese da Classificação dos Problemas.

NÍVEL	CLASSE DO PROBLEMA	DESCRIÇÃO
1	**Superficial**	Causa desconforto ao usuário, porém não compromete a execução das ações, exigindo-lhe apenas um processo de adaptação. Tal fato não implica que o problema de usabilidade não deva ser solucionado, partindo do pressuposto que não é o usuário quem deve se "amoldar" às características do produto e, sim, o inverso (e.g., falta de clareza em mensagens de erro, ausência de realimentação no processo interativo).
	Intermediário	Causa desconforto ao usuário, além de forçá-lo a alterar o curso de suas ações, e.g., uma falha no acesso de uma opção ou sub-opção do menu de uma interface multimodal (menus, linguagem de comandos e manipulação direta) destinados a diferentes categorias de usuários.
	Grave	Causa grande desconforto ao usuário, por comprometer seriamente a execução das ações, e.g., travamento de uma função que exija o reinício do processo (*parcial*) ou de todo o sistema (*total*).
2	**De consistência**	Relativo a conflitos entre partes do sistema, tanto em nível estrutural e estético, quanto semântico e operacional, e.g., apresentação de mensagens de erro referentes à mesma função ou funções afins em diferentes regiões da tela em cores distintas (estrutura e estética) e/ou apresentação de mensagens de erro semanticamente divergentes relativas à mesma função ou funções afins (semântica e operacional).
	Recorrente	Interfere no processo interativo a cada vez que se repetem determinadas operações, e.g., a "quebra" do retorno de informações ao usuário cada vez que este solicita a aquisição de parâmetros estatísticos relativos a uma imagem analisada em um sistema de processamento de imagens.
	Geral	Afeta várias partes do sistema, e.g., falhas que induzem panes parciais ou totais no sistema.

Fonte: Adaptado de [Queiroz, 2001].

A fim de evidenciar a importância da condução dos ensaios em ambientes distintos, e.g., laboratório e campo, pode-se realizar comparações relacionadas à natureza e ao número de falhas encontrados em cada um dos ambientes.

Subetapa 25: Triangulação dos dados

Consiste na confrontação das três categorias de dados coletados, i.e., os dados relacionados à *mensuração do desempenho*, *sondagem da satisfação* e *inspeção de conformidade* foram confrontados, de maneira a detectar problemas adicionais não evidenciados durante a triagem isolada dos dados obtidos a partir de cada enfoque considerado, além de inconsistências nos dados.

Exemplo 04: Síntese de Resultados

A Triagem Preliminar dos Dados consistiu na totalização de tempos de execução de tarefas, de erros repetidos, de opções e ações incorretas, de consultas a diferentes mecanismos de ajuda, comentários verbais, opiniões, falhas detectadas na inspeção dos produtos-alvo e o de ações incorretas de reconhecimento de voz. Em seguida, foi levada a efeito a análise dos dados selecionados, de forma a detectar problemas não identificados diretamente durante as sessões de teste. Os principais produtos gerados a partir da Tabulação e Síntese dos Dados podem ser visualizados na Tabela 1 e no Quadro 18. Para apresentar os dados relacionados ao perfil dos usuários, recomenda-se o uso de gráficos (Figura 12).

Tabela 1 – Taxas de Adoção no *HP iPAQ 910c*.

PADRÃO	#P	#S	TA(%)
ISO 9241 Part 14	64,0	69,0	92,75%
ISO 9241 Part 16	32,0	34,0	94,12%
ISO 9241 Part 17	54,0	57,0	94,74%
ISO 14754	6,0*	11,0	54,54%
ISO 24755	10,0**	17,0	58,82%

P — Recomendações adotadas pelo produto
S — Recomendações aplicáveis ao produto avaliado
TA — Taxa de adoção

Quadro 18 – Síntese da mensuração da satisfação dos usuários com o *HP iPAQ 910c*.

Mensuração da Satisfação dos Usuários com o *HP iPAQ 910c*	
Laboratório	Campo
0,39	0,36
Bastante Satisfeito	Bastante Satisfeito
Escala de satisfação Subjetiva	
Intervalo	Descrição
0,67 a 1,00	Satisfação Máxima
0,33 a 0,66	Bastante Satisfeito
0,01 a 0,32	Pouco Satisfeito
0,00	Neutro
-0,01 a -0,32	Pouco Insatisfeito
-0,33 a -0,66	Bastante Insatisfeito
-0,67 a -1,00	Insatisfação Máxima

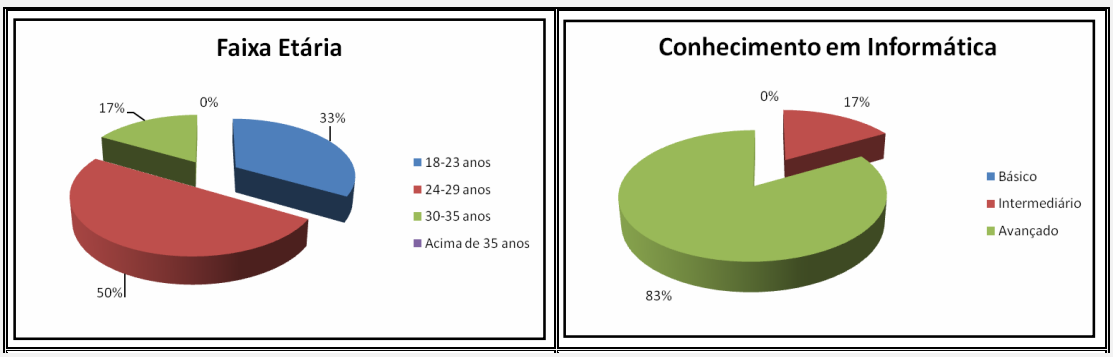


Figura 12 – Excerto do *Delineamento do Perfil dos Usuários*.

Um excerto dos problemas encontrados a partir da inspeção de conformidade e da mensuração do desempenho é apresentado, respectivamente, nos Quadros 19 e 20. Por fim, um excerto da triangulação dos dados é exibido no Quadro 21.

Quadro 19 – Falhas e Parecer sobre o *HP iPAQ 910c* com base na inspeção de conformidade.

PADRÃO	# DA FALHA	DESCRIÇÃO
ISO 9241 Part 14	01	As opções de *menu* mais importantes devem ser listadas primeiro, assim a sub-opção *New* deveria ser a primeira opção do painel *menu* nas aplicações *Excel Mobile* e *Word Mobile.*
	02	Presença de um ícone sem função na aplicação *Calendar.*
ISO 14754	03	O comando via reconhecimento de escrita para exclusão não foi adotado.
	04	O comando via reconhecimento de escrita para mover não foi adotado.
ISO 24755	05	Ícone de configurações com projeto distinto do que recomendado no padrão.
	06	Ícone de aplicação de vídeo com projeto distinto do que recomendado no padrão.
PARECER: Considerando os padrões utilizados, constata-se que, apesar de algumas falhas não comprometer de forma significativa o processo de execução das tarefas, tais falhas necessitam ser corrigidas, a fim de que o processo de interação seja melhorado e torne-se mais ágil par o usuário.		

Quadro 20 – Sumário das falhas detectadas na mensuração do desempenho do *HP iPAQ 910c*.

ID	Descrição
01	Ausência de uma indicação visual mais nítida para o botão ligar.
02	Dificuldade para retirar a caneta *stylus.*
03	Controle de volume está distante de sua indicação visual.
04	Ausência de indicação visual para a caneta *stylus.*
05	Após uso prolongado, dispositivo apresenta concentração de calor.
06	Ausência de destaque visual no local de inserção dos dados de reconhecimento de escrita.
PARECER: Apesar da maioria das falhas serem superficiais, e não ter sido detectada nenhuma falha grave, as correções precisam ser implementadas de maneira a não comprometer o processo de interação do usuário com o produto.	

Quadro 21 – Confronto dos resultados obtidos a partir dos diferentes enfoques de avaliação no *HP iPAQ 910c*.

ID	NÍVEL GENÉRICO	NÍVEL ESPECÍFICO	IC	MD	SS
01	**Localização e seqüência das Opções de menu**	As opções de *menu* mais importantes devem ser listadas primeiro, assim a sub-opção *New* deveria ser a primeira opção do painel *menu* nas aplicações *Excel Mobile* e *Word Mobile.*	*V*		*X*
		O painel menu associado a *One Note Mobile* não mantém a ordem convencional das sub-opções. A opção *Rename* deveria ser lista primeiro que a opção *Delete.*	*V*		
02	**Navegação por menus**	Presença de um ícone sem função na aplicação *Calendar.*	*V*		
		Não há consistência entre o ícone *Calendar* na tela *Today* e o utilizado no teclado.	*V*	*V*	
03	**Voz**	Há palavras utilizadas para a seleção de opções por voz que não são foneticamente distintas.	*V*		*V*
		O comando "*Compose email to"* não é reconhecido facilmente, porém a abreviação do comando para "*email to"* é aceito		*V*	

1.3.6 Elaboração do Relatório de Avaliação

A etapa final do processo de avaliação compreende a definição do modo de divulgação dos resultados obtidos e a elaboração do documento final que contém a apresentação e discussão dos resultados da avaliação.

1.4 Considerações Finais

Uma análise dos dados coletados nas avaliações de produtos que seguiram a metodologia ora apresentada permite evidenciar que determinadas categorias de problemas de usabilidade são mais facilmente detectadas por determinada dimensão ou técnica de avaliação, conforme pode ser visualizado no Quadro 21 do Exemplo 04.

Além disso, os dados advindos das avaliações laboratoriais e de campo não têm se mostrado divergentes, mas complementares. Reforçando, assim, a relevância de uma abordagem metodológica multidimensional para a avaliação da usabilidade de interfaces para aplicações móveis e multimodais.

Referências

Baille, L., and Schatz, R. (2005) "Exploring Multimodality in the Laboratory and the Field", In: International Conference On Multimodal Interfaces (ICMI'2005), Trento, Italy. Proc. New York: ACM Press, 2005. p. 100-107.

Balbo, S., Coutaz, J. and Salber, D. (2003) "Towards automatic evaluation of multimodal user interfaces", Intelligent User Interfaces, Knowledge-Based Systems, v. 6, no. 4, p. 267–274.

Barbosa, A. E. V. (2009) "Abordagem Híbrida para a Avaliação de Interfaces Multimodais", Dissertação (Mestrado em Ciência da Computação) – Pós-graduação em Informática, Universidade Federal de Campina Grande, Campina Grande.

Bernhaupt, R.; Navarre, D.; Palanque, P.; Winckler, M. (2007) "Model-Based Evaluation: A New Way to Support Usability Evaluation of Multimodal Interactive Applications", In: Law, E.; Hvannberg, E.; Cockton, G. Maturing Usability: Quality in Software, Interaction and Value. Berlin: Springer-Verlag, pp. 96-119, Human-Computer Interaction Series.

Bias, R. G. and Mayhew, D. J., Cost-justifying usability. Boston, MA: Academic Press, 1994.

Blumendorf, M., Feuerstack, S. and Albayrak, S. (2008) "Multimodal Smart Home User Interfaces", New York: ACM Press, Disponível em: <http://www.uni-koblenz.de/~confesc/IUI/proceedings/data/workshops/w2_04.pdf>. Acessado em: maio 2009.

Bosert, J. L., Quality Functional Deployment: A Practitioner's Approach. ASQC Quality Press, New York, 1991.

Bowman, D., Gabbard, J. and Hix, D. (2002) "A survey of usability evaluation in virtual environments: Classification and comparison of methods", Presence: Teleoperators and Virtual Environment, v. 11, no. 4, p. 404-424.

Brewster, S. A., Wright, P. C., and Edwards, A. D. N. (1994) "The design and evaluation of an auditory-enhanced scrollbar", In: Conference On Human Factors In Computing Systems (CHI´94). Proc. New York: ACM Press, pp. 173-179.

Croasmun, J. (2004) “Are Ergonomists Really Consulted in Mobile Phone Design?” Ergoweb, 17 Jul. 2004, Disponível em: <http://www.ergoweb. com/news/detail.cfm?id=961>. Acesso em: 04 Jun. 2005.

Dillon, R. F., Edey, J. D., and Tombaugh, J. W. (1990) “Measuring the true cost of command selection: techniques and results”, In J. C. Chew & J. Whiteside (Eds.), ACM Conference on Human Factors in Computing Systems (CHI´90), Seattle, Washington, ACM Press, pp. 19-25.

Dix, A. J., Finlay, J. E., Abowd, G. D., Beale, R. Human-Computer Interaction. 3. ed. Upper Saddle River, NJ: Prentice-Hall, 2003.

Duda, S., Schiel, M., and Hess, J. M. (2002) “Mobile Usability”. Empfehlungen für die Entwicklung benutzerfreundlicher mobiler Datendienste.

Dybkjær, L., Bernsen, N. O. and Minker, W. (2004a) “Evaluation and usability of multimodal spoken language dialogue systems”, Speech Communication, 2004, v. 43, pp. 33-54.

Dybkjær, L., Bernsen, N. O., and Minker, W. (2004b) “New challenges in usability evaluation – Beyond task-oriented spoken dialogue systems”, In: International Conference On Spoken Language Processing (ICSLP 2004), Jeju Island, Korea, v. 3, p. 2261–2264.

Ferreira, D. S. (2007) “Abordagem Híbrida para a Avaliação da Usabilidade de Dispositivos Móveis”, Dissertação (Mestrado em Informática) – Pós-graduação em Informática, Universidade Federal de Campina Grande, Campina Grande.

Grant, J. (2004) “Developing usability principles and guidelines for web content on PDAs”. SIMS Research Students’ Colloquium 2004.

Gorlenko, L. and Merrick, R. (2003) “No wires attached: Usability challenges in the connected mobile world”, IBM Systems Journal, vol. 42, n. 4, p. 639-651.

Hagen, P., Robertson, T., Kan, M. and Sadler, K. (2005) “Emerging research methods for understanding mobile technology use”, In Proc. of OZCHI 2005, Australia.

Hinckley, K., Pausch, R., Proffitt, D. and Kassel, N. F. (1998) “Two-handed virtual manipulation”, ACM Trans. on Computer-Human Interaction, v. 5, no. 3, p. 260-302.

HP – HP iPAQ 910c Business Messenger. 2009. Disponível em: <http://h10010.www1.hp.com/wwpc/br/pt/ho/WF05a/215348-215348-64929-3352590-3352590-3551665.html>. Acessado em: Jun. 2009.

Inácio Jr., V. R. (2007) “Um framework para desenvolvimento de interfaces multimodais em aplicações de computação ubíqua”, Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) – Universidade de São Paulo, São Carlos, 2007.

ISO – International Organization for Standardization. ISO 9241 – Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) – *Part 1: General introduction*. Suécia, 2001.

ISO – International Organization for Standardization. ISO 9241 Ergonomics requirements for office work with visual displays terminals (VDTs) - *Part 11: Guidance on usability*. International Standard. Suécia, 1998a.

ISO – International Organization for Standardization. ISO 9241 – Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) – *Part 14: Menu Dialogues*. Geneva, 1997.

ISO – International Organization for Standardization. ISO 9241 – Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) – *Part 16: Direct manipulation dialogues*. Geneva, 1999a.

ISO – International Organization for Standardization. ISO 9241 – Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) – *Part 17: Form filling dialogues*. Geneva, 1998b.

ISO – International Organization for Standardization. ISO 14754 – Pen- Based Interfaces – Common gestures for Text Editing with Pen- Based Systems. Geneva, 1999b.

ISO – International Organization for Standardization. ISO 18021 – User interfaces for mobile tools for management of database communications in a client-server model. Geneva, 2002.

ISO – International Organization for Standardization. ISO 24755 – Screen icons and symbols for personal mobile communication device. Geneva, 2006b.

ITU – International Telecommunication Union. *E.161 - Arrangement of digits, letters and symbols on telephones and other devices that can be used for gaining access to a telephone network*. 2001.

Jöst, M., Haubler, J., Merdes, M., and Malaka, R. (2005) "Multimodal interaction for pedestrians: an evaluation study", In: International Conference On Intelligent User Interfaces (IUI'2005), San Diego, California, USA. Proc. New York: ACM Press. p. 59-66.

Kamba, T.; Elson, S. A.; Harpold, T.; Stamper, T. Sukaviriya, P. N. (1996) "Using small screen space more efficiently", In Proc. of CHI'96, April, p.383-390.

Kaster, T., Pfeiffer, M. and Bauckhage, C. (2003) "Combining speech and haptics for intuitive and efficient navigation through image database". In: S. Oviatt (Ed.), International Conference On Multimodal Interfaces (ICMI'2003), Vancouver, British Columbia, Canada. Proc. New York: ACM Press. p. 180-187.

Ketola, P. and Röykkee, M. (2002) "The three facets of usability in mobile handsets", Disponível em: <http://www.cs.colorado.edu/~palen/chi_workshop/papers/ketola.pdf>, Acessado em: 06 Jun. 2005.

Kjeldskov, J. and Stage, J. (2004) "New techniques for usability evaluation of mobile systems", International Journal on Human-Computer Studies, v. 60, no. 5, pp. 599-620.

Kjeldskov, J. (2002) "Just-in-place Information for Mobile Device Interfaces", In Proc. of the 4th International Symposium on Mobile Human-Computer Interaction, p. 271-275.

Klein, A., Schwank, I., Genereux, M. and Trost, H. (2001) "Evaluating Multi-modal Input Modes in a Wizard-of-Oz Study for the Domain of Web Search" In: Conference On People And Computers (HCI´01), Lille, France. Proc. Springer, 2001. pp. 475-484.

Klockar, D.; Carr, D. A.; Hedman, A.; Johansson, T. Bengtsson, F. (2003) "Usability of mobile phones", In Proc. of the 19th International Symposium on Human Factors in Telecommunications, December, p. 197-204.

Lai, J. (2004) "Facilitating Mobile Communication with Multimodal Access to Email Messages on a Cell Phone", In Proc. of CHI'04. ACM Press, New York, pp. 1259-1262.

Levine, D. M., Berenson, M. L.; Stephan, D. 2000 EstatíStica: Teoria E AplicaçõEs - Usando Microsoft Excel PortuguêS. LTC.

Mallick, M. Mobile and Wireless Design Essentials, Canada: Wiley Publishing, Inc., 2003.

Maybury, M. (2001) "Coordination and Fusion in Multimodal Interaction", Disponível em:<http://www.mitre.org/work/tech_papers/tech_papers_01/maybury_coordination/maybury_coordination.pdf >, Acessado em: jan. 2008.

Nedel, L., Freitas, C. M. D. S., Jacob, L. and Pimenta, M. (2003) "Testing the use of egocentric interactive techniques in immersive virtual environments", In: M. Rauterberg, M. Menozzi & J. Wesson (Eds.), IFIP Conference on Human-Computer Interaction (INTERACT'2003), Zurich, Switzerland. IOS Press, p. 471-478.

Nielsen, J., Usability Engineering, Cambridge: Academic Press, 1993.

Oliveira, R. C. L. de. (2005) "WebQuest: Uma Ferramenta Web Configurável para a Sondagem da Satisfação Subjetiva do Usuário", Dissertação (Mestrado em Informática) – Pós-graduação em Informática, Universidade Federal de Campina Grande, Campina Grande.

Oviatt, S. L. (2003) "Multimodal Interfaces", In: JACKO, J.; SEARS, A. (Eds.). Handbook of Human-Computer Interaction. New Jersey: Lawrence Erlbaum, pp. 286-304.

Paternò, F. and Santos, I. (2006) "Designing and developing multi-user, multi-device web interfaces", In G. Calvary & J. Vanderdonckt (Eds.), Conference On Computer-Aided Design Of User Interfaces (CADUI2006), Bucharest. Proc. Springer Verlag, 2006. pp. 111-122.

Petridis, P., Mania, K., Pletinckx, D. and White, M. (2006) "Usability evaluation of the EPOCH multimodal user interface: designing 3D tangible interactions", In: Symposium On Virtual Reality Software And Technology (VRST´06), 2006, Limassol, Cyprus. Proc. New York: ACM Press, 2006. pp. 116-122.

Poupyrev, I., Weghorst, S., Billinghurst, M. and Ichikawa, T. (1998) "Egocentric object manipulation in virtual environments: empirical evaluation of interaction techniques", In N. Ferreira & M. Göbel (Eds.), Proc. of Computer Graphics Forum (EUROGRAPHICS'98). Malden, MA: Blackwell Publishers, p. 41-52.

Preece, J., Rogers, Y. and Sharp, H., Design de Interação: além da interação homem-computador, Porto Alegre: Bookman, 2005.

Queiroz, J. E. R. De; Oliveira, R. C. L. De; Turnell. M. F. Q. V. (2005) "WebQuest: A Configurable Web Tool to Prospect the User Profile and User Subjective Satisfaction". In: HCI International 2005. Las Vegas, Nevada. July 22-27. Volume 2 - The Man. of Information: E-Business, the Web, and Mobile Computing.

Queiroz, J. E. R. (2001) "Abordagem Híbrida para a Avaliação da Usabilidade de Interfaces com o Usuário", Tese (Doutorado em Engenharia Elétrica) – Pós-graduação em

Engenharia Elétrica, Universidade Federal da Paraíba, Campina Grande.

Reis, T., Sá, M. and Carriço, L. (2008) "Multimodal Artefact Manipulation: Evaluation in Real Contexts", 3rd International Conference On Pervasive Computing And Applications, Alexandria, Egypt. Marwan Al-Akaidi, Lizhen Cui, Bin Hu, Bo Hu, Zongkai Lin, Yong Zhang (Eds.), Proc. IEEE Press, 2008. pp. 570-575.

Rosson, M. B. and Carroll, J. M., Usability Engineering: Scenario-based development of human-computer interaction, San Francisco: Academic Press, 2002.

Schapira, E. and Sharma, R. (2001) "Experimental Evaluation of Vision and Speech based Multimodal Interfaces", In: Workshop On Perceptive User Interfaces, 2001, Orlando, Florida. Proc. New York: ACM Digital Library, 2001. pp. 1-9.

Seffah, A., and Metzker E., Adoption-centric Usability Engineering: Systematic Deployment, Assessment, and Improvement of Usability Methods in Software Engineering, Springer, 2009.

Shackel, B., Human Factors for Informatics Usability, University Press, Cambridge, 1991.

Stanciulescu, A., Vanderdonckt, J. and Macq, B. (2007) "Automatic Usability Assessment of Multimodal User Interfaces Based on Ergonomic Rules". Praud, S. (Ed.), Disponível em: <http://www.isys.ucl.ac.be/bchi/publications/2007/Stanciulescu-EMODE2007.pdf>.

St. Amant, R.; Horton, T. E.; Ritter, F. E. (2004) "Model-based evaluation of cell phone menu interaction", In Proc. of the ACM Conference on Human Factors in Computing Systems (CHI'04), April, p. 343-350.

Taib, R. and Ruiz, N. (2006) "Multimodal interaction styles for hypermedia adaption", In: International Conference On Intelligent User Interfaces (IUI´06), 2006, Sydney, Australia. Proc. New York: ACM Press, 2006. pp. 351-353.

Tidwell, J., Designing Interfaces, O'Reilly Media, Inc., USA, 2005.

Trevisan, D. G., Nedel, L. P., Macq, B. and Vanderdonckt, J. (2006) "Detecting interaction variables in a mixed reality system for maxillofacial-guided surgery", In: SBC Symposium On Virtual Reality (SRV´2006), 2006, Belém. Proc. SBC Press, 2006. pp. 39-50.

Wahlster, W. Dialogue systems go multimodal: the SmartKom experience. In: WAHLSTER, W. (ed.), SmartKom: foundations of multimodal dialogue systems. Secaucus, NJ, USA: Springer, 2006. pp. 3-27.

WebQuest – Portal do WebQuest. 2009. Disponível em: <http://webquest.paqtc.org.br>. Acesso em: 18 Jun. 2009.

Weiser, M. (1991) "Some computer science problems in ubiquitous computing", In: Communications of the ACM 36, n. 7, p. 75-84.

Weiss, S. Handheld Usability, New York: John Willey & Sons, Ltd., 2002.

Zhu, M. A speech recognition interface for PDAs and cell phones. Drexel University, Philadelphia, 2004.

Capítulo 2

Redes Sociais Online: Técnicas de Coleta e Abordagens de Medição

Fabrício Benevenuto
Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Belo Horizonte - Brasil
fabricio@dcc.ufmg.br

Resumo

Redes sociais online se tornaram extremamente populares e vêm causando uma nova onda de aplicações na Web. Associado a esse crescimento, redes sociais estão se tornando um tema central em pesquisas de diversas áreas. Este trabalho oferece uma introdução ao pesquisador que pretende explorar esse tema. Inicialmente, apresentamos as principais características das redes sociais mais populares atualmente. Em seguida, discutimos as principais métricas e tipos de análises utilizadas no estudo dos grafos que formam a topologia das redes sociais. Finalmente, sumarizamos as principais abordagens utilizadas para se obter dados de redes sociais online e discutimos trabalhos recentes que utilizaram essas técnicas.

Abstract

Online social networks became extremely popular and are increasingly causing a new wave of applications on the Web. Associated to this popularity growth, online social networks are becoming a key theme in several research areas. This work offers an introduction to the researcher that aims at exploring this theme. Initially, we present the main characteristics of current online social network sites. Then, we discuss the main metrics and types of analysis used to study the graphs that represent the social network topologies. Finally, we summarize the main approaches used to collect online social networks and discuss recent work that used these approaches.

2.1. Introdução

Desde seu início, a Internet tem sido palco de uma série de novas aplicações, incluindo a Web, aplicações par-a-par e email. Atualmente, a Web vem experimentando

uma nova onda de aplicações associada à proliferação das redes sociais e ao crescimento de mídia digital. Várias redes sociais online (OSNs - *Online Social Networks*) surgiram, incluindo redes de profissionais (ex., LinkedIn), redes de amigos (ex., MySpace, Facebook, Orkut), e redes para o compartilhamento de conteúdos específicos tais como mensagens curtas (ex., Twitter), diários e blogs (ex., LiveJournal), fotos (ex., Flickr), e vídeos (ex., YouTube).

Redes sociais online têm atraído milhões de usuários. De acordo com a *Nielsen Online* [80], mídia social passou na frente de email como a atividade online mais popular. Mais de dois terços da população online global visita ou participa de redes sociais e blogs. Como comparação, se o Facebook fosse um país, seus 500 milhões de usuários registrados colocariam o Facebook como terceiro país mais populoso do mundo [5]. Tanta popularidade está associada a uma funcionalidade comum de todas as redes sociais online que é permitir que usuários criem e compartilhem conteúdo nesses ambientes. Este conteúdo pode variar de simples mensagens de texto comunicando eventos do dia-a-dia até mesmo a conteúdo multimídia, como fotos e vídeos. Como conseqüência, as estatísticas sobre conteúdo gerado pelos usuários nesses sítios Web são impressionantes. O Facebook compartilha mais de 60 bilhões de fotos, que ocupam mais de 1.5 PB de espaço [10]. A quantidade de conteúdo que o YouTube armazena em 60 dias seria equivalente ao conteúdo televisionado em 60 anos, sem interrupção, pelas emissoras NBC, CBS e ABC juntas [12]. De fato, o YouTube foi acessado por mais de 100 milhões de usuários apenas em Janeiro de 2009 [1], com uma taxa de upload de 10 horas de vídeo por minuto [14].

Apesar de tanta popularidade e da enorme quantidade de conteúdo disponível, o estudo de redes sociais ainda está em sua infância, já que esses ambientes estão experimentando novas tendências e enfrentando diversos novos problemas e desafios. A seguir sumarizamos alguns elementos motivadores para o estudo de redes sociais online.

- **Comercial**: Com usuários passando muito tempo navegando em redes sociais online, esses sítios Web tem se tornado um grande alvo para propagandas. De fato, em 2007, 1,2 bilhões de dólares foram gastos em propagandas em redes sociais online no mundo todo, e é esperado que este número triplique até 2011 [21]. Além disso, redes sociais online são lugares onde usuários compartilham e recebem uma grande quantidade de informação, influenciando e sendo influenciado por amigos [38]. Conseqüentemente, redes sociais online estão se tornando cada vez mais um alvo de campanhas políticas [51] e de diversas outras formas de marketing viral, onde usuários são encorajados a compartilhar anúncios sobre marcas e produtos com seus amigos [64].

- **Sociológica**: No passado o estudo de redes sociais era um domínio de sociólogos e antropólogos, quando ferramentas típicas para se obter dados eram entrevistas [88]. Como conseqüência, muitos desses esforços foram realizados baseados em amostras de dados pequenas e pouco representativas. Com o surgimento de redes sociais online, surgiu a oportunidade de estudos nesse tema com o uso de grandes bases de dados. Sistemas como Facebook, Twitter, Orkut, MySpace e YouTube possuem milhões de usuários registrados e bilhões de elos que os conectam. Redes sociais permitem o registro em larga escala de diversos aspectos da sociologia e da natureza humana relacionados à comunicação e ao comportamento humano. Além

disso, redes sociais online vem funcionando como um novo meio de comunicação e modificando aspectos de nossas vidas. Redes sociais online permitem que as pessoas interajam mais, permite que pessoas mantenham contato com amigos e conhecidos e permitem indivíduos se expressar e serem ouvidas por uma audiência local ou até mesmo global.

- **Melhorias dos sistemas atuais**: Assim como qualquer sistema Web, redes sociais online são vulneráveis a novas tendências e estão sujeitas a verem seus usuários rapidamente se mudar para outros sistema sem aviso prévio. Por exemplo, o MySpace experimentou um crescimento exponencial no número de usuários seguido de uma forte queda depois de abril de 2008 devido a um aumento no número de usuários do Facebook [85]. No início, o Orkut cresceu rapidamente em diversos lugares, mas sua popularidade foi concretizada somente em alguns países, dos quais o Brasil é o país com maior número de usuários registrados [11]. Várias razões podem explicar este tipo de fenômeno, incluindo a interface e novas utilidades do sistema, problemas de desempenho e características dos usuários, etc. Finalmente, o grande volume de dados disponível em redes sociais online abre um novo leque para a pesquisa relacionada à recuperação de conteúdo, onde estratégias de busca e recomendação de usuários e conteúdo são cada vez mais importante.

  Outro aspecto importante está relacionado ao tráfego gerado pelas redes sociais online. Intuitivamente, existe uma diferença crucial entre publicar conteúdo na Web tradicional e compartilhar conteúdo através de redes sociais online. Quando as pessoas publicam algum conteúdo na Web, elas tipicamente fazem isso para que todos os usuários da Internet, em qualquer lugar, possam acessar. Por outro lado, quando usuários publicam conteúdo em redes sociais online, eles geralmente possuem uma audiência em mente, geralmente, seus amigos. Algumas vezes, a audiência é explicitamente definida por um usuário ou pela política do sistema. Conseqüentemente, redes sociais online constituem uma classe única de aplicações com potencial para remodelar o tráfego da Internet com sua popularidade crescente. Estudar aspectos de sistemas relacionados a redes sociais pode ser de grande importância para a próxima geração da infra-estrutura da Internet e sistemas de distribuição de conteúdo [59, 81].

- **Segurança e conteúdo indesejável**: Redes sociais estão cada vez mais se tornando alvo de usuários maliciosos ou oportunistas que enviam propagandas não solicitadas, spam, e até mesmo *phishing*. O problema se manifesta de diversas maneiras, como postagens em listas de vídeos mais populares contendo spam [29, 26], spam no Twitter [24], conteúdo com metadados que não descrevem o conteúdo [27], etc. Conteúdo não solicitado consome a atenção humana, talvez o recurso mais importante na era da informação. O ruído e o distúrbio causados por alguns usuários reduzem a efetividade da comunicação online e é um problema cada vez maior.

Redes sociais compõem ambientes perfeitos para o estudo de vários temas da computação, incluindo sistemas multimídia e interação humano-computador. Além disso, por permitir que usuários criem conteúdo, redes sociais vêm se tornando um tema chave em

pesquisas relacionadas à organização e tratamento de grandes quantidades de dados, além de constituírem um ambiente ideal para extração de conhecimento e aplicação de técnicas de mineração de dados. Neste mini-curso apresentamos uma visão geral sobre redes sociais, oferecendo uma base necessária ao pesquisador que pretende explorar o tema. Inicialmente, apresentamos as principais características das redes sociais mais populares atualmente. Em seguida, discutimos as principais métricas e tipos de análises utilizadas no estudo dos grafos que formam a topologia das redes sociais. Finalmente, sumarizamos as principais abordagens utilizadas para se obter dados de redes sociais online e discutimos trabalhos recentes que utilizaram essas técnicas.

2.2. Definições e Características de Redes Sociais Online

Esta seção apresenta uma visão geral sobre as redes sociais online, suas características em comum e os principais mecanismos de interação entre os usuários.

2.2.1. Definição

O termo rede social online é geralmente utilizado para descrever um grupo de pessoas que interagem primariamente através de qualquer mídia de comunicação. Conseqüentemente, baseado nessa definição, redes sociais online existem desde a criação da Internet. Entretanto, neste trabalho, nós utilizaremos uma definição um pouco mais restrita, adotada em trabalhos anteriores [33, 69]. Nós definimos uma rede social online como um serviço Web que permite indivíduos (1) construir perfis públicos ou semi-públicos dentro de um sistema, (2) articular uma lista de outros usuários com os quais compartilham conexões e (3) visualizar e percorrer suas listas de conexões e outras listas feitas por outros no sistema.

Com base nessa definição existem várias redes sociais online disponíveis na Web, que variam de acordo com seus propósitos. A tabela 2.1 sumariza os propósitos de várias redes sociais online populares. Uma lista atualizada e exaustiva de redes sociais online, com mais de 150 sítios Web, pode ser encontrada em [8].

Nome	**Propósito**	**URL**
Orkut	Amizades	http://www.orkut.com
Facebook	Amizades	http://www.facebook.com
MySpace	Amizades	http://www.myspace.com
Hi5	Amizades	http://www.hi5.com
LinkedIn	Professionais	http://www.linkedin.com
YouTube	Compartilhamento de vídeos	http://www.youtube.com
Flickr	Compartilhamento de fotos	http://www.flickr.com
LiveJournal	Blogs e diários	http://www.livejournal.com
Digg	Compartilhamento de (*bookmarks*)	http://digg.com
Twitter	Troca de mensagens curtas	http://twitter.com
Last FM	Compartilhamento de rádio/músicas	http://www.last.fm

Tabela 2.1. Algumas redes sociais online populares

2.2.2. Elementos das redes sociais online

A seguir discutimos várias funcionalidades oferecidas pelas redes sociais atuais. O objetivo desta seção não é prover uma lista completa e exaustiva de funcionalidades, mas apenas descrever as mais relevantes.

- **Perfis dos usuários**: Redes sociais online possuem muitas funcionalidades organizadas ao redor do perfil do usuário, na forma de uma página individual, que oferece a descrição de um membro. Perfis podem ser utilizados não só para identificar o indivíduo no sistema, mas também para identificar pessoas com interesses em comum e articular novas relações. Tipicamente, perfis contêm detalhes demográficos (idade, sexo, localização, etc.), interesses (passatempos, bandas favoritas, etc.), e uma foto. Além da adição de texto, imagens e outros objetos criados pelo usuário, o perfil na rede social também contém mensagens de outros membros e listas de pessoas identificadas como amigos na rede. Perfis são geralmente acessíveis por qualquer um que tenha uma conta na rede social online ou podem ser privados, de acordo com as políticas de privacidades definidas pelo usuário.

  Recentemente, Boyd e colaboradores [32] mostraram que para a maior parte dos usuários de redes sociais online, existe uma forte relação entre a identidade do indivíduo real e seu perfil na rede social.

- **Atualizações**: Atualizações são formas efetivas de ajudar usuários a descobrir conteúdo. Para encorajar usuários a compartilhar conteúdo e navegar por conteúdo compartilhado por amigos, redes sociais online geralmente fazem as atualizações imediatamente visíveis aos amigos na rede social. Burke e colaboradores [37] mostram que atualizações motivam contribuições de novos usuários no sistema. Eles conduziram um estudo utilizando dados de 140,000 novos usuários do Facebook para determinar que atividades como atualizações são vitais para que novos usuários contribuam para o sistema. Como atualizações podem receber comentários de outros usuários, atualizações também são formas especiais de comunicação em redes sociais online.

- **Comentários**: A maior parte dos sítios de redes sociais permite que usuários comentem em conteúdo compartilhado. Alguns sistemas também permitem que usuários comentem em outros perfis de usuários. Comentários são um meio primordial de comunicação em redes sociais online, que também expressam relações sociais [19, 44]. Como exemplo, vídeos podem receber comentários no YouTube, fotos podem receber comentários no Facebook, Flickr e Orkut, usuários do LiveJournal podem postar comentários em blogs, etc.

- **Avaliações**: Em muitas redes sociais online, o conteúdo compartilhado por um usuário pode ser avaliado por outros usuários. Avaliações podem aparecer em diferentes níveis de granularidade e formas. No Facebook, usuários podem apenas gostar de uma postagem, clicando no botão "*I like this*". Com mais de 500 milhões de usuários registrados, cada usuário do Facebook avalia 9 objetos a cada mês em média [5]. No YouTube, vídeos podem ser avaliados com até 5 estrelas, de forma similar à avaliação empregada na categorização de hotéis. O YouTube ainda provê

uma avaliação binária (positiva ou negativa) para os comentários recebidos por vídeos, na tentativa de filtrar comentários ofensivos ou com alguma forma de spam.

Avaliações de conteúdo são úteis de várias formas. Como exemplo, elas são importantes para sistemas como o YouTube para ajudar usuários a encontrar e identificar conteúdo relevante. Avaliações podem ainda ajudar administradores a identificar conteúdo de baixa qualidade ou mesmo conteúdo inapropriado. Além disso, avaliações podem ser utilizadas para outras finalidades no sistema, como conteúdo em destaque, sistemas de recomendação, etc. Uma rede social online que coloca as avaliações dos usuários no centro do sistema é o Digg. O Digg permite que usuários avaliem URLs, notícias ou estórias e utiliza aquelas mais votadas para expor o conteúdo mais popular [63].

- **Listas de Favoritos**: Várias aplicações sociais utilizam listas de favoritos para permitir usuários selecionar e organizar conteúdo. Listas de favoritos ajudam usuários a gerenciar seu próprio conteúdo e podem ser úteis para recomendações sociais. Como exemplo, usuários podem manter listas de vídeos favoritos no YouTube e fotos favoritas no Flickr. Nesses sistemas, usuários podem navegar na lista de favoritos de outros usuários para buscar novos conteúdos [40]. Conseqüentemente, listas de favoritos também funcionam como uma forma de descoberta de conteúdo e propagação de informação. Sistemas como o Orkut e o Twitter também provêem listas de favoritos (fãs).

- **Listas de Top**: Tipicamente, redes sociais que colocam algum tipo de conteúdo de mídia como elemento central do sistema, como o YouTube, provêem listas de conteúdo mais popular ou usuários mais populares. Geralmente, essas listas são baseadas em avaliações ou outras estatísticas do sistema relativas ao conteúdo (ex. número de visualizações, avaliações, número de comentários) ou relativas aos usuários (ex. número de assinantes).

- **Metadados**: Uma das novas tendências da Web 2.0 é permitir aos usuários criar conteúdo livremente e associar metadados ao conteúdo [45]. Em redes sociais online como o YouTube e o Flickr, usuários tipicamente associam metadados como título, descrição e tags ao conteúdo compartilhado. Metadados são essenciais para recuperação de conteúdo em redes sociais online.

2.3. Teoria de redes complexas

Redes sociais online são inerentemente redes complexas e vários estudos vêm estudando características de redes sociais online utilizando como base teorias existentes [15, 70, 46, 60, 65, 23, 28]. De fato, o estudo de redes complexas cobre um grande número de áreas e sua teoria tem sido utilizada como ferramenta para entender vários fenômenos, incluindo o espalhamento de epidemias [71], propagação de informação [90], busca na Web [36], e consequências de ataques a redes de computadores [17]. A seguir, várias propriedades estatísticas e métricas comumente utilizadas para analizar e classificar rede complexas são apresentadas na seção 2.3.1. As seções 2.3.2 e 2.3.3 discutem propriedades de redes small-world e redes power-law. Uma revisão detalhada sobre métricas e teoria de redes complexas pode ser encontrada na referência [76].

2.3.1. Métricas para o estudo de redes

Uma rede é um conjunto de ítens, que chamamos de vértices ou nodos, com conexões entre si, chamadas de arestas. Em outras palavras, uma rede nada mais é do que um grafo. Existem diversas propriedades estatísticas e métricas que caracterizam a estrutura dessas redes, discutidas a seguir. Assume-se que o leitor tenha um conhecimento sobre a terminologia utilizada em teoria de grafos.

2.3.1.1. Grau dos vértices

Uma característica importante sobre a estrutura de uma rede é a distribuição dos graus de seus vértices. Consequentemente, uma métrica comum utilizada para comparar redes é o expoente α obtido através da regressão linear de uma distribuição de lei de potência. Valores típicos para o expoente α ficam entre 1.0 e 3.5 [49]. Para redes direcionadas, é comum analisar o grau dos nodos em ambas as direções, o grau de entrada e o grau de saída.

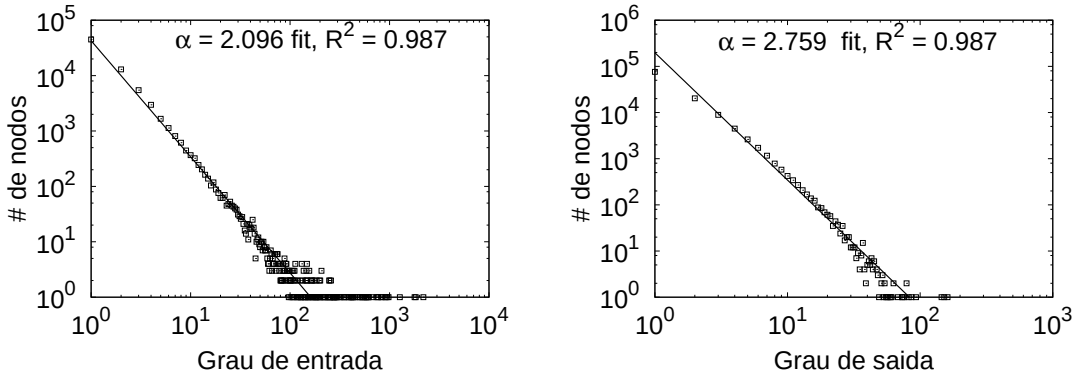


Figura 2.1. Grau de entrada e de saída de um grafo de interações entre usuários através de vídeos do YouTube

Como exemplo, a figura 2.1 mostra a distribuição dos graus de entrada e saída para um grafo formado de interações entre usuários de vídeos do YouTube, utilizado nas referências [23, 28]. Note que a curva da regressão linear, utilizada para se calcular o expoente α também é exibida nesses gráficos. Ferramentas como o Gnuplot [6] e o Matlab [9] podem ser utilizados para se realizar a regressão e calcular o valor de α. Para verificar a acurácia da regressão, é comum medir o fator R^2 [86], sendo que se o valor de R^2 for 1, significa que não há diferenças entre o modelo e os dados reais.

2.3.1.2. Coeficiente de clusterização

O coeficiente de clusterização de um nodo i, $cc(i)$, é a razão entre do número de arestas existentes entre os vizinhos de um nodo e o total de arestas possíveis entre os vizinhos de i. Como exemplo a Figura 2.2 mostra o valor do coeficiente de clusterização para o nodo escuro em três cenários diferentes. No primeiro, todos os vizinhos do nodo estão conectados entre si e, conseqüentemente, o cc do nodo é 1. No segundo cenário,

existe apenas 1 aresta entre os vizinhos do nodo dentre as 3 possíveis, deixando o nodo com cc=1/3. No último cenário, não há nenhuma aresta entre os vizinhos do nodo escuro e, portanto, o cc do nodo é 0.

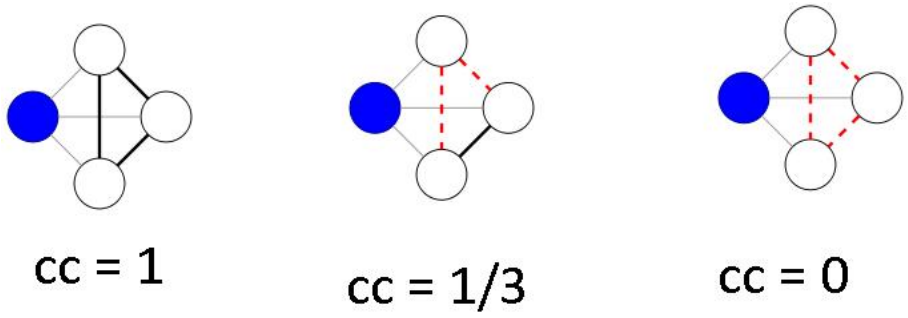


Figura 2.2. Cálculo do coeficiente de clusterização de um nodo em três cenários diferentes

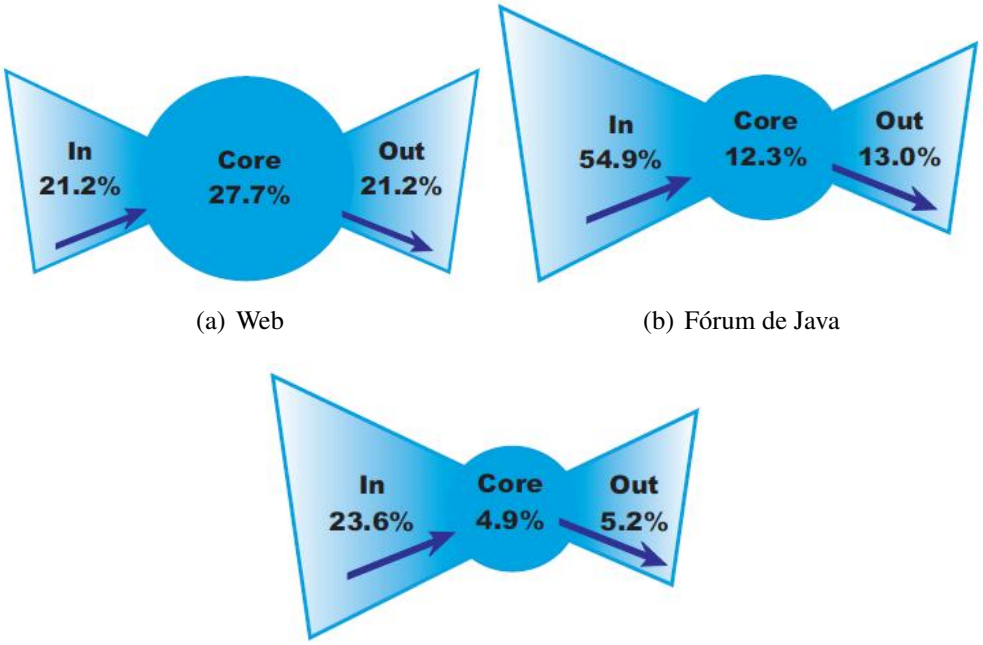
Podemos notar que o coeficiente de clusterização funciona como uma medida da densidade de arestas estabelecidas entre os vizinhos de um nodo. O coeficiente de clusterização de uma rede, *CC*, é a média do coeficiente de clusterização de todos os nodos.

2.3.1.3. Componentes

Um componente em um grafo é um conjunto de nodos, onde cada nodo possui um caminho para todos os outros nodos do conjunto. Um componente é chamado de fortemente conectado (SCC - *Strongly Connected Component*) quando o caminho entre todos os nodos do conjunto é direcionado. Por outro lado, dizemos que um componente é fracamente conectado (WCC - *Weakly Connected Component*) se o caminho é não direcionado.

Um trabalho que se tornou referência no estudo de componentes em redes complexas aborda a estrutura da Web (nodos são páginas Web e arestas são elos existentes entre as páginas) [36]. Os autores propõem um modelo que representa como os componentes no grafo da Web se relacionam. Este modelo, aplicado somente em grafos direcionados, possui um componente central que é o SCC, chamado também de *core*, e outros grupos de componentes que podem alcançar o SCC ou serem alcançados por ele. O modelo ficou conhecido como *bow tie* [36], pois a figura que ilustrada o modelo lembra uma gravata borboleta.

Este modelo tem sido utilizado por outros estudos como forma de comparar a organização dos componentes de um grafo direcionado [93, 28]. A figura 2.3 compara a estrutura dos componentes de três diferentes grafos utilizando-se o modelo *bow tie*. O componente central, *core*, das figuras corresponde à fração dos nodos do grafo que fazem parte do SCC. O componente *in* contém os nodos que apontam para algum nodo do *core*, mas não são apontados por nodos desse componente. Finalmente, o componente *out* corresponde aos nodos que são apontados por nodos do *core*.



(a) Web

(b) Fórum de Java

(c) Interações com vídeo

Figura 2.3. Estrutura dos componentes da Web [36], do Fórum de Java [93] e do grafo de interações através de vídeos [28]

2.3.1.4. Distância média e diâmetro

A distância média de um grafo é o número médio de passos entre todos os caminhos mínimos existentes para todos os nodos do grafo. Normalmente, a distância média é computada apenas no SCC para grafos direcionados ou no WCC para grafos não direcionados, já que não existe caminho entre nodos localizados em componentes diferentes. Outra métrica relacionada é o diâmetro do grafo. O diâmetro é definido como a distância do maior caminho mínimo existente no grafo (em geral, também computado somente no WCC ou no SCC).

2.3.1.5. Assortatividade

De acordo com Newman [75], assortatividade é uma medida típica de redes sociais. Uma rede exibe propriedades assortativas quando nodos com muitas conexões tendem a se conectar a outros nodos com muitas conexões. Sendo assim, definimos como $knn(k)$ como o grau médio de todos os vizinhos dos nodos com grau k. A assortatividade ou disassortatividade de uma rede é geralmente avaliando plotando-se o $knn(k)$ em função de k.

2.3.1.6. Betweenness

O Betweeness é uma medida relacionada à centralidade dos nodos ou de arestas na rede. O betweeness B de uma aresta é definido como o número de caminhos mínimos entre todos os pares de nodos em um grafo que passam pela aresta [78]. Se entre um par de nodos possui múltiplos caminhos mínimos entre eles, cada caminho recebe um peso de forma que a soma dos pesos de todos os caminhos seja 1. Conseqüentemente, o betweenness de uma aresta e pode ser expressado como

$$B(e) = \sum_{u \in V, v \in V} \frac{\sigma_e(u,v)}{\sigma(u,v)} \tag{1}$$

onde $\sigma(u,v)$ representa o número de caminhos mínimos entre u e v, e $\sigma_e(u,v)$ representa o número de caminhos mínimos entre u e v que incluem e. O betweenness de uma aresta indica a importância dessa aresta no grafo em termos de sua localização. Arestas com maior betweenness estão em mais caminhos mínimos e, portanto, são mais importantes para a estrutura do grafo.

De forma similar, o betweenness pode ser computado para um nodo ao invés de uma aresta. Neste caso, a medida do betweenness mede o número de caminhos mínimos que passam por nodo. Nodos que possuem muitos caminhos mínimos que passam por eles possuem maior betweenness do que os outros que não possuem.

2.3.1.7. Reciprocidade

Uma forma interessante de se observar a reciprocidade em um grafo direcionado é medindo a probabilidade de um nodo ter uma aresta apontando para ele para cada nodo que ele aponta. Em outras palavras, a reciprocidade ($R(x)$) é dada por:

$$R(x) = \frac{|O(x) \cap I(x)|}{|O(x)|} \tag{2}$$

onde $O(x)$ é o conjunto de nodos que recebem uma aresta de um usuário x e $I(x)$ é o conjunto de nodos que apontam x através de arestas direcionadas.

Outra métrica interessante de ser observar é o coeficiente de reciprocidade ρ, uma métrica que captura a reciprocidade da interações em toda a rede [52]. O coeficiente de reciprocidade ρ é definido pelo coeficiente de correlação entre entidades de uma matriz de adjacência de um grafo direcionado ($a_{ij} = 1$ if há uma aresta de i para j, senão $a_{ij} = 0$):

$$\rho = \frac{\sum_{i \neq j}(a_{ij} - \bar{a})(a_{ji} - \bar{a})}{\sum_{i \neq j}(a_{ij} - \bar{a})^2}, \tag{3}$$

onde o valor médio $\bar{a} = \sum_{i \neq j}(a_{ij}/N(N-1)$ e N é o número de usuários no grafo.

O coeficiente de reciprocidade indica se o número de arestas recíprocas na rede é maior ou menor do que o de uma rede aleatória. Se o valor ρ é maior do que 0, a rede é recíproca; caso contrário, anti-recíproca.

2.3.1.8. PageRank

O PageRank é um algoritmo interativo que assinala um peso numérico para cada nodo, com o propósito de medir sua importância relativa dentro dos nodos do grafo. O algoritmo foi inicialmente proposto por Brin and Page [35] para ordenar resultados de busca do protótipo do Google. A intuição por trás do PageRank é que uma página Web é importante se existem muitas páginas apontando para ela ou se existem páginas importantes apontando para ela. A equação que calcula o PageRank (PR) de um nodo é definida da seguinte forma:

$$PR(u) = (1-d) + d \sum_{v \in B(u)} \frac{PR(v)}{N_v} \tag{4}$$

onde u representa um nodo. $B(u)$ é é o conjunto de páginas que apontam para u. $PR(u)$ e $PR(v)$ são os valores do PageRank para os nodos u e v, respectivamente. N_v denomina o número de arestas que saem do nodo v, e o parâmetro d é um fator que pode ter valor entre 0 e 1.

O algoritmo do PageRank tem sido aplicado em outros contextos, como por exemplo, para encontrar usuários experientes em fóruns especializados [93] e usuários influentes no Twitter [91, 61]. Além disso, existem outras modificações do PageRank com propósitos específicos, como por exemplo, a detecção de spam na Web [56].

2.3.2. Redes small-world

O conceito de redes small-world ficou bastante conhecido com o famoso experimento de Milgram [68]. Seu experimento consistiu de um grupo de voluntários tentando enviar uma carta para uma pessoa alvo através de outras pessoas que eles conheciam. Milgram enviou cartas a várias pessoas. As cartas explicavam que ele estava tentando atingir uma pessoa específica final em uma cidade dos EUA e que o destinatário deveria repassar a carta para alguém que ele achasse que poderia levar a carta o mais próximo do seu destino final (ou entregá-la diretamente, caso o destinatário final fosse uma pessoa conhecida). Antes de enviar a carta, entretanto, o remetente adicionava seu nome ao fim da carta, para que Milgram pudesse registrar o caminho percorrido pela carta. Das cartas que chegaram com sucesso ao destino final, o número médio de passos requerido para o alvo foi 6, resultado que ficou conhecido como o princípio dos *seis graus de separação*.

Em termos das propriedades das redes sociais que discutimos, uma rede pode ser considerada small-world se ela tiver duas propriedades básicas: coeficiente de clusterização alto e um pequeno diâmetro [89]. Estas propriedades foram verificadas em várias redes como a Web [18, 36], redes de colaboração científica [74, 77] (pesquisadores são nodos e arestas ligam co-autores de artigos), atores de filmes [20] (atores são nodos e arestas ligam atores que participaram do mesmo filme), e redes sociais on-

line [15, 70, 46, 65, 23, 28]. Em particular, Mislove e colaboradores [70] verificaram propriedades small-world em quatro redes sociais online: LiveJournal, Flickr, Orkut, e YouTube.

2.3.3. Redes power-law

Redes power-law consistem de grafos cuja distribuição de grau segue uma distribuição de lei de potência. Em outras palavras, nessas redes a probabilidade que um nodo tenha grau k é proporcional a $k^{-\alpha}$ para $\alpha > 1$. Várias redes reais mostram distribuições de grau que seguem distribuições de lei de potência, incluindo a topologia da Internet [50], a Web [22], e redes neurais [34].

Redes livres de escala (scale free) são classes de redes que seguem leis de potência, onde os nodos de grau alto tendem a se conectar a outros nodos de grau alto. Barabási e colaboradores [22] propuseram um modelo para gerar redes livre de escala, introduzindo o conceito de conexão preferencial (*preferential attachment*). O modelo diz que a probabilidade de um nodo se conectar a outro nodo é proporcional ao seu grau. Os autores do modelo ainda mostraram que, sob certas circunstâncias, este modelo produz redes que seguem leis de potência. Mais recentemente, Li e colaboradores [66] criaram uma métrica para medir se uma rede é livre de escala ou não, além de prover uma longa discussão sobre o tema.

2.4. Técnicas de Coleta de Dados

Em um passado recente, redes sociais eram um domínio de sociólogos e antropólogos, quando ferramentas típicas de coleta de dados de redes sociais eram pesquisas e entrevistas com pequenos grupos de usuários [88]. Com o surgimento das redes sociais online, a obtenção desse tipo de dados em larga escala se tornou possível e diversas áreas da computação começaram a realizar coletas de dados.

Diferentes áreas de pesquisa demandam diferentes tipos de dados e, por isso, existem várias formas de se obter dados de redes sociais online. A figura 2.4 apresenta possíveis pontos de coleta de dados, que variam desde entrevistas com os usuários até à instalação de coletores localizados em servidores Proxy ou aplicações. A seguir discutimos essas diferentes abordagens, bem como trabalhos que adotaram essas estratégias.

2.4.1. Dados dos usuários

Um método comum de se analisar o uso de redes sociais online consiste em conduzir entrevistas com usuários desses sistemas. Em particular, esta estratégia tem sido bastante empregada pela comunidade da área de interface homem-máquina [84, 57, 41, 31, 79], onde entrevistas estruturadas são as formas mais populares de obtenção de dados.

Como exemplo, através de entrevistas com usuários do Facebook, Joinson e seus colaboradores [57] identificaram várias razões pelas quais usuários utilizam o Facebook, incluindo conexão social, compartilhamento de interesses, compartilhamento e recuperação de conteúdo, navegação na rede social e atualização do seu estado atual. Chapman e Lahav [41] conduziram entrevistas e analisaram a navegação de 36 usuários de quatro nacionalidades diferentes para examinar diferenças etnográficas no uso de redes sociais online.

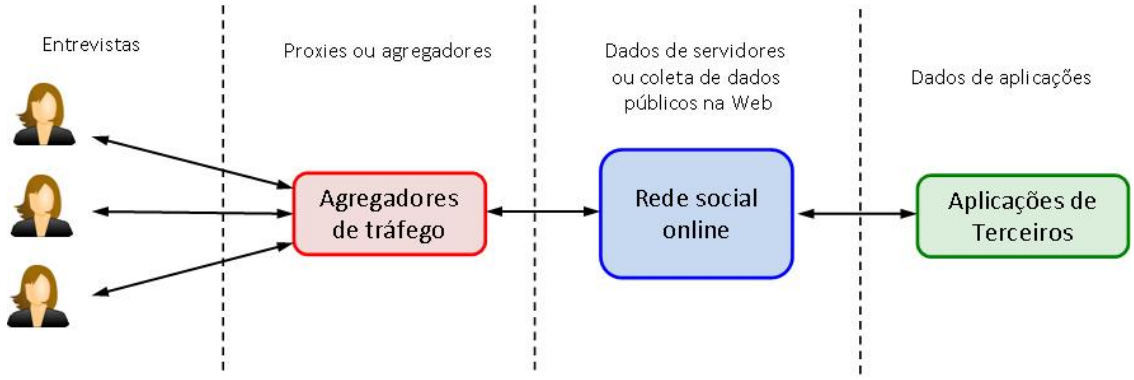


Figura 2.4. Possíveis pontos de coleta de dados

2.4.2. Dados de pontos intermediários

Existem duas técnicas comuns utilizadas para coletar dados de pontos de agregação de tráfego na rede. A primeira consiste em coletar os dados que passam por um provedor de serviços Internet (ISP) e filtrar as requisições que correspondem a acessos às rede sociais online. A segunda consiste em coletar dados diretamente de um agregador de redes sociais. A seguir, discutimos alguns trabalhos que fizeram o uso dessas estratégias.

2.4.2.1. Servidores proxy

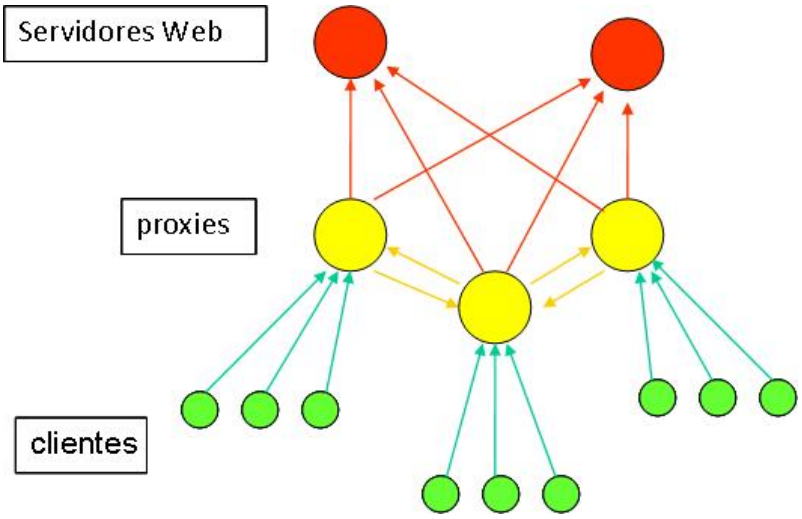


Figura 2.5. Exemplo de um servidor proxy intermediando o tráfego entre clientes e servidores

Coletar dados de um servidor proxy tem sido uma estratégia utilizada em vários estudos sobre o tráfego da Internet [47, 55, 67, 92]. A figura 2.5 ilustra como um servidor funciona como um agregador de tráfego de seus clientes. Tais servidores são utilizados para delimitar uma porção da rede, onde computadores estão localizados em uma mesma localização geográficas.

Dado o crescente interesse por vídeos na Web, alguns trabalhos recentes utilizaram

servidores proxy para obter dados do tráfego gerado por sistemas como o YouTube. Gill e colaboradores [53] caracterizaram o tráfego do YouTube coletando dados de um servidor proxy localizado na universidade de Calgary, no Canadá. Eles mostraram que requisições de HTTP GET, utilizadas para baixar conteúdo do servidor, correspondem a 99.87% do total das requisições que passam pelo servidor proxy. Eles ainda caracterizaram diversas medidas tais como a duração, a idade e a categoria dos vídeos. Mais recentemente, os mesmos autores caracterizam sessões no YouTube [54]. Eles mostraram que uma sessão típica possui cerca de 40 minutos e utilizaram esse valor para analisar medidas relativas à sessão dos usuários, tais como duração da sessão, tempo entre sessões e tipos de conteúdo transferidos em cada sessão. Finalmente, Zink e colaboradores [94] também estudaram tráfego coletado no proxy de uma universidade. Eles caracterizaram medidas típicos no tráfego do YouTube e, utilizando essas medidas, eles simularam abordagens de caches locais e globais para vídeos, bem como o uso de uma arquitetura P2P para distribuição de vídeos. De maneira geral, eles mostraram que essas abordagens poderiam reduzir tráfego significativamente e permitir acesso aos vídeos de forma mais rápida.

Em um trabalho recente, Schneider e colaboradores [82] extraíram dados de redes sociais online de um provedor de acesso a Internet e reconstruíram ações realizadas pelos usuários em suas navegações por diferentes redes sociais online. Em outras palavras, eles criam o que chamamos de *clickstream* [42] para redes sociais online, que captura cada passo da navegação dos usuários do ISP. Eles oferecem uma ampla discussão sobre a metodologia para reconstruir os acessos dos usuários e, com base nesses dados, eles analisaram as seqüências de requisições realizadas pelos usuários em redes sociais online como o Facebook.

2.4.2.2. Agregador de redes sociais

Agregadores de redes sociais são sistemas que permitem acesso a várias redes sociais simultaneamente, através de um portal único. Esses sistemas ajudam usuários que utilizam várias redes sociais online a gerenciar vários perfis de uma forma mais simples e unificada [58, 83]. Ao logar em um agregador de redes sociais online, usuários acessam suas contas através de uma interface única, sem precisar logar em cada rede social separadamente. Isto é feito através de uma conexão HTTP em tempo real realizada em duas etapas. A primeira etapa ocorre entre o usuário e o agregador de redes sociais e a segunda etapa ocorre entre o sistema agregador e as redes sociais. Agregadores tipicamente comunicam com redes sociais online através de APIs, como o OpenSocial [7], e todo o conteúdo é exibido através da interface do sistema agregador. A figura 2.6 descreve o esquema de interação entre os usuários, um sistema agregador e algumas redes sociais online. Através dessa interface, um usuário pode utilizar várias funcionalidades de cada rede social que ele está conectado, como checar atualizações de amigos, enviar mensagens e compartilhar fotos.

Recentente, nós utilizamos esta estratégia para obter dados de *clickstream* de redes sociais online [30]. Nós colaboramos com um agregador de redes sociais online e obtivemos dados da navegação dos usuários em 4 redes sociais online: Orkut, Hi5, MySpace e LinkedIn. A tabela 2.2 mostra o número de usuários, sessões e requisições HTTP para

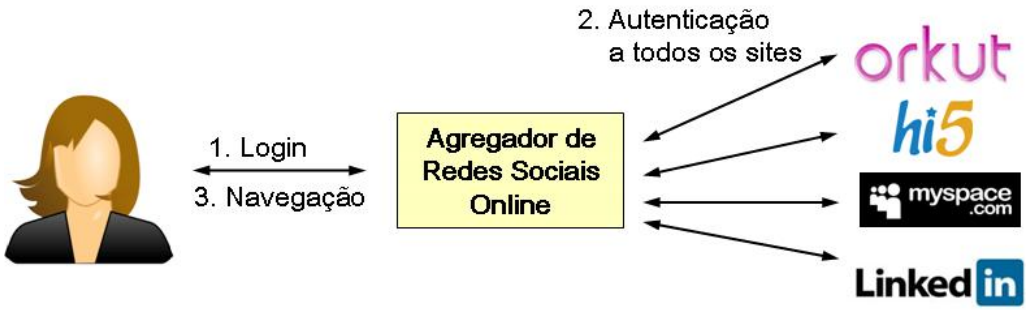


Figura 2.6. Ilustração de um usuário se conectando a múltiplas redes sociais online simultaneamente através de um portal agregador

cada uma dessas redes. Baseados nesses dados e em dados coletados do Orkut, nós examinamos o comportamento dos usuários nas redes sociais online, bem como características das interações entre os usuários através das várias atividades que eles realizam.

	# usuários	# sessões	# requisições
Orkut	36.309	57.927	787.276
Hi5	515	723	14.532
MySpace	115	119	542
LinkedIn	85	91	224
Total	37.024	58.860	802.574

Tabela 2.2. Sumário dos dados obtidos de um agregador de redes sociais

2.4.3. Dados de servidores de redes sociais online

Idealmente, servidores de redes sociais são os locais mais adequados para a coleta de dados. Entretanto, a maior parte desses sistemas evita prover dados, mesmo que anonimizados. Existem alguns poucos trabalhos que utilizaram dados obtidos diretamente de servidores de uma rede social. Chun e seus colaboradores [44] estudaram interações textuais entre os usuários do Cyworld, uma rede social bastante popular na Coréia do Sul, através de dados obtidos diretamente do servidor. Eles compararam a rede de amizades explicita com a rede criada por mensagens trocadas no livro de visitas do Cyworld, discutindo diversas similaridades e diferenças em termos da estrutura da rede. Baluja e colaboradores obtiveram dados dos servidores do YouTube. Eles utilizaram dados do histórico da navegação dos usuários do YouTube para criar um grafo onde cada vídeo é um nodo e arestas ligam vídeos freqüentemente vistos em seqüência. Baseados nesse grafo, eles criaram um mecanismo capaz de prover sugestões de vídeo personalizadas para os usuários do YouTube. Finalmente, Duarte e seus colaboradores [48] caracterizaram o tráfego em um servidor de blogs do UOL (www.uol.com.br). Recentemente, nós estudamos a navegação dos usuários em um servidor de vídeos do UOL, chamado UOL Mais [25].

Dada a dificuldade em se obter dados diretamente de servidores de redes sociais online, uma estratégia comum consiste em visitar páginas de redes sociais com o uso de uma ferramenta automática, que chamamos de *crawler* ou robô, e coletar sistematicamente informações públicas de usuários e objetos. Tipicamente, os elos entre usuários de

uma rede social online podem ser coletados automaticamente, permitindo que os grafos de conexões entre os usuários sejam reconstruídos. Essa estratégia tem sido amplamente utilizada em uma grande variedade de trabalhos, incluindo estudos sobre a topologia das redes sociais online [70, 16], padrões de acesso no YouTube [39] e interações reconstruídas através de mensagens trocadas pelos usuários [87]. A seguir discutimos vários aspectos relacionados à coleta de dados de redes sociais online.

2.4.3.1. Coleta por amostragem

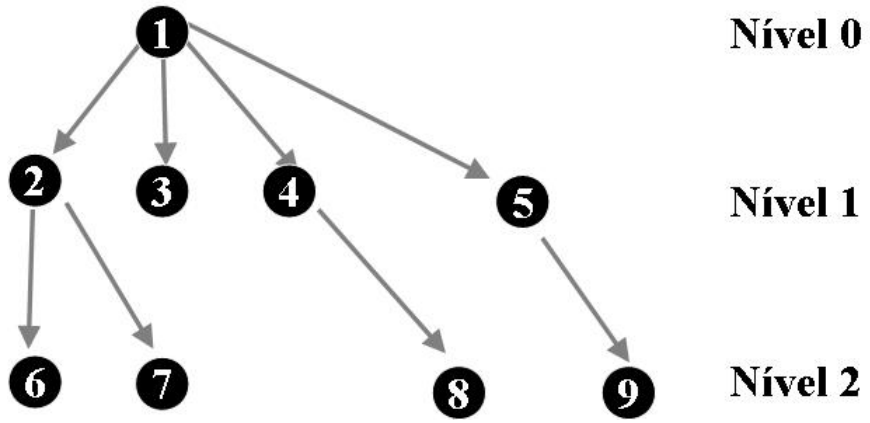


Figura 2.7. Exemplo de busca em largura em um grafo

Idealmente, é sempre mais interessante coletar o grafo inteiro de uma rede social online para evitar que a coleta seja tendenciosa a um grupo de usuários da rede. Entretanto, na maior parte das vezes, não há uma forma sistemática de se coletar todos os usuários de uma rede social online. Para esses casos é necessário coletar apenas parte do grafo. Uma abordagem comumente utilizada chama-se snowball. Snowball consiste em coletar o grafo de uma rede social online seguindo uma abordagem de busca em largura, como ilustra a figura 2.7. A coleta inicia-se a partir de nodo semente. Ao coletar a lista de vizinhos desse nodo, novos nodos são descobertos e então coletados no segundo passo, que só termina quando todos os nodos descobertos no primeiro passo são coletados. No passo seguinte todos os nodos descobertos no passo anterior são coletados, e assim sucessivamente. Recomenda-se o uso de um grande número de nodos sementes para evitar que a coleta não fique restrita a um pequeno componente do grafo.

O que caracteriza a coleta por snowball é a interrupção da coleta em um passo intermediário, antes que todos os nodos alcançáveis pela busca em largura sejam atingidos. Dependendo do objetivo da coleta, snowball pode ser uma estratégia viável. Por exemplo, se realizarmos 3 passos da coleta por snowball, podemos calcular o coeficiente de clusterização dos nodos semente. Entretanto, se quisermos computar o coeficiente de clusterização médio de toda a rede ou outras métricas como distribuição de graus, distância média, etc., a coleta por snowball pode resultar em números tendenciosos [62, 16].

Outra abordagem bastante difundida consiste em coletar o maior componente fracamente conectado (WCC) do grafo. A coleta de um componente inteiro pode ser realizada com uma estratégia baseada em um esquema de busca em largura ou busca em

profundidade. Quanto maior o número de sementes utilizadas maior a chance de se coletar o maior componente do grafo. Em trabalhos recentes, nós realizamos uma busca por palavras aleatórias no YouTube para verificar se o componente coletado era o maior componente [28, 23]. Como a maior parte dos usuários encontrados nessas buscas estavam no WCC do nosso grafo, os resultados desse teste sugeriram que o componente coletado era o maior WCC. Mislove e colaboradores [70] argumenta que o maior WCC de um grafo é estruturamente a parte mais interessante de ser analisada, pois é o componente que registra a maior parte das atividades dos usuários. Além disso, eles mostram que usuários não incluídos no maior WCC tendem fazer parte de um grande grupo de pequenos componentes isolados ou até mesmo totalmente desconectados.

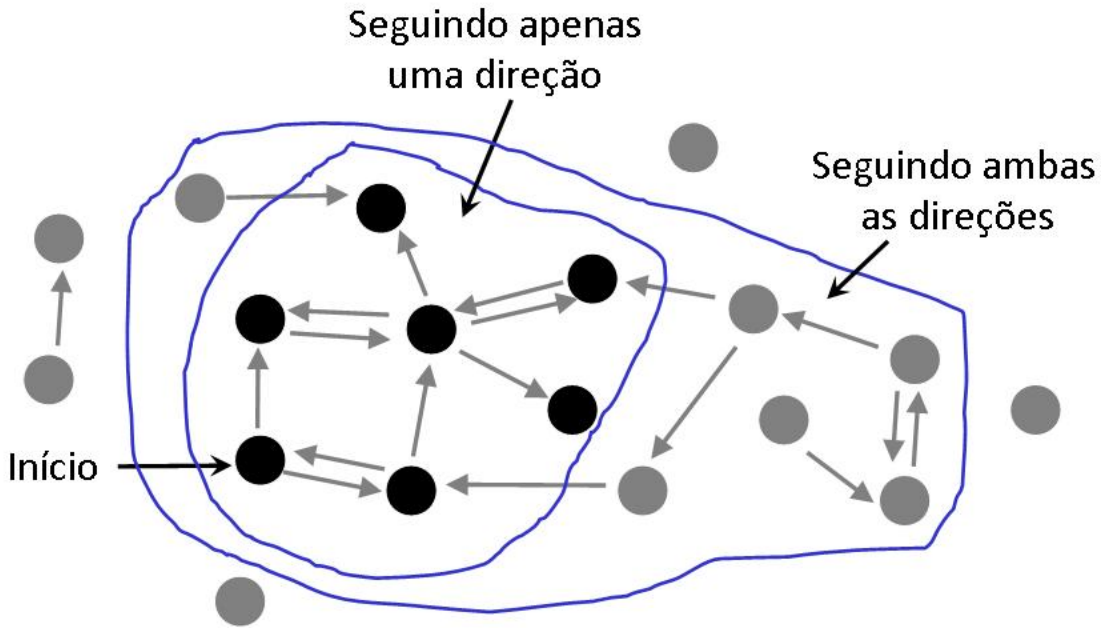


Figura 2.8. Exemplo de coleta do WCC em um grafo direcionado

Finalmente, é importante observar que para coletar o WCC em um grafo direcionado é necessário percorrer as arestas do grafo em ambas as direções. Algumas redes sociais online como o Twitter ou o Flickr, o conceito de amizade é direcionado. Ou seja, um usuário pode seguir outro, mas não ser seguido pelo mesmo. Se coletarmos o grafo seguindo as arestas em apenas uma direção, não necessariamente vamos coletar todo o WCC. A figura 2.8 mostra que o conjunto de nodos coletados quando seguimos as arestas em ambas as direções é maior do que quando seguimos apenas uma das direções. Em algumas redes não é possível percorrer o grafo em ambas as direções e, portanto, não é possível coletar o maior WCC. Essa limitação é típica de estudos que envolvem a coleta da Web [36]. Tipicamente, a Web é freqüentemente coletada seguindo apenas uma direção os elos entre as páginas, já que não é possível determinar o conjunto de páginas que apontam para uma página.

2.4.3.2. Coleta em larga escala

A coleta de grandes bases de dados de redes sociais online geralmente envolve o uso de coletores distribuídos em diversas máquinas. Isso acontece não só devido ao processamento necessário para tratar e salvar os dados coletados, mas também para evitar que servidores de redes sociais interpretem a coleta de dados públicos como um ataque a seus servidores.

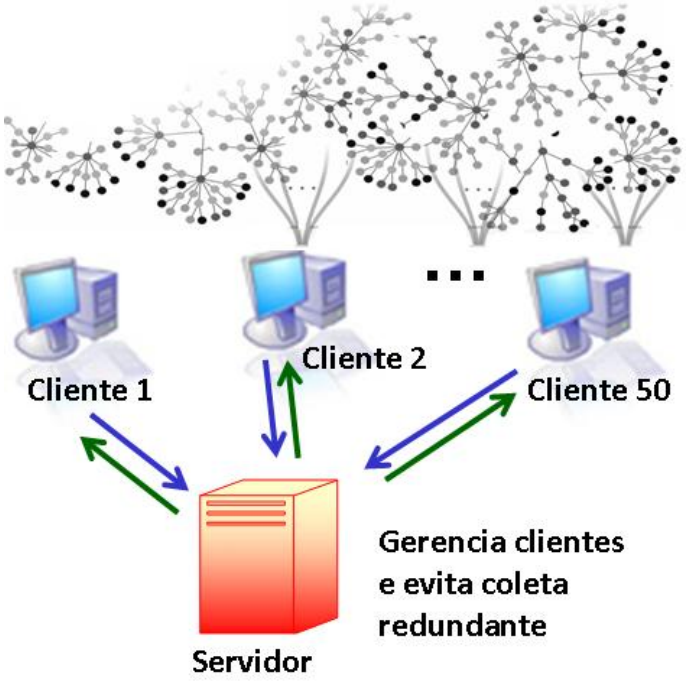


Figura 2.9. Exemplo de coleta feita de forma distribuída

Uma forma de se realizar tal coleta, conforme descrito em [43], está representada na figura 2.9. A estratégia consiste em utilizar (1) uma máquina mestre que mantém uma lista centralizada de usuários a serem visitados e (2) máquinas escravas, que coletam os dados, salvam esses dados, fazem um parser dos dados coletados e identificam novos usuários. Finalmente, as máquinas escravas retornam listas de novos usuários identificados à máquina mestre, que por sua vez, distribui novos usuários para as máquinas escravas.

2.4.3.3. Coleta por inspeção de IDs

Como discutido anteriormente, ao coletar uma rede social online o ideal é coletar toda a rede e não só uma porção dela. Em alguns sistemas como o MySpace e o Twitter é possível realizar uma coleta completa. Esses sistemas atribuem um identificador (ID) numérico e seqüencial para cada usuário cadastrado. Como novos usuários recebem um identificador seqüencial, podemos simplesmente percorrer todos os IDs, sem ter que verificar a lista de amigos desses usuários em busca de novos IDs para coletar.

Recentemente, nós realizamos uma coleta do Twitter seguindo essa estratégia. Nós pedimos aos administradores do Twitter para realizar uma coleta em larga escalae eles adicionaram 58 IPs de nossas máquinas em uma lista banca, com permissão para coletar. Cada uma das 58 máquinas, localizadas no *Max Planck Institute for Software Systems (MPI-SWS)*, na Alemanha[1], teve permissão para realizar uma taxa máxima de 20 mil requisições por hora ao Twitter. Utilizando a API do Twitter, nosso coletor investigou todos 80 milhões de IDs de forma seqüência, coletando todas as informações públicas sobre os usuários, bem como seus elos de seguidores e seguidos e todos os seus tweets. Dos 80 milhões de contas inspecionadas, nós encontramos cerca 55 milhões em uso. Isso acontece porque o Twitter apaga contas inativas por um período maior do que 6 meses. No total, coletamos cerca de 55 milhões de usuários, quase 2 bilhões de elos sociais e cerca de 1.8 bilhões de tweets. Ao inspecionar as listas de seguidores e seguidos coletadas, nós não encontramos nenhum ID acima do 80 milhões inspecionados, sugerindo que nós coletamos todos os usuários. Esses dados foram utilizados recentemente em dois trabalhos, um sobre detecção de spammers no Twitter [24] e o outro sobre medição de influência no Twitter [38].

Torkjazi e colaboradores [85] também usufruíram dos IDs sequênciais do MySpace para inspecionar o surgimento de novos usuários no sistema.

2.4.3.4. Utilizando APIs

Figura 2.10. Exemplo da API do Twitter: http://twitter.com/users/show/fbenevenuto.xml

No contexto de desenvolvimento Web, uma API é tipicamente um conjunto de tipos de requisições HTTP juntamente com suas respectivas definições de resposta. Em redes sociais é comum encontrarmos APIs que listam os amigos de um usuário, seus objetos, suas comunidades, etc. APIs são uma nova tendência na Web 2.0 onde sistemas não só oferecem seus serviços, mas permitem que outros acessem dados através de APIs.

APIs são perfeitas para a coleta de dados de redes sociais, pois oferecem os dados em formatos estruturados como XML e JSON. Vários sistemas como YouTube e Twitter

[1]Esta coleta foi realizada durante uma visita de 5 meses ao MPI-SWS

oferecem APIs. Como exemplo, a figura 2.10 mostra o resultado de uma busca por informações de um usuário no Twitter. Além dessa função, o Twitter ainda oferece várias outras funções em sua API. Com a API do Twitter é possível coletar 5000 seguidores de um usuário através de uma única requisição. A coleta dessa informação através do sítio Web convencional necessitaria centenas de requisições, visto que o Twitter só mostra alguns seguidores por página. Além disso, cada página viria com uma quantidade muito maior de informação desnecessária.

Vários sistemas possuem APIs, incluindo Twitter, Flickr, YouTube, Google Mapas, Yahoo Mapas, etc. Com tantas APIs existentes, é comum ver aplicações que utilizam duas ou mais APIs para criar um novo serviço, que é o que chamamos de Mashup. Uma interessante aplicação chamada Yahoo! Pipes [13], permite a combinação de diferentes APIs de vários sistemas para a criação automatizada de Mashups.

2.4.3.5. Ferramentas e bibliotecas

```
#!/usr/bin/perl

use LWP;

$ua = LWP::UserAgent->new();
$req = new HTTP::Request(GET =>
   "http://twitter.com/followers/ids/44446416.xml?page=1");
$content = $ua->request($req)->content;

print "$content";
```

Figura 2.11. Exemplo do uso da biblioteca LWP em Perl

Desenvolver um crawler pode ser uma tarefa bastante complicada devido à diversidade de formatos de páginas. Entretanto, coletar redes sociais online é, em geral, diferente de coletar páginas da Web tradicional. As páginas de uma rede social online são, em geral, bem estruturadas e possuem o mesmo formato, pois são geradas automaticamente, enquanto na Web tradicional as páginas podem ser criadas por qualquer pessoa em qualquer formato. Além disso, como cada indivíduo ou objeto em uma rede social, em geral, possui um identificador único, temos certeza sobre quais as informações obtivemos quando coletamos uma página.

```
#!/usr/bin/python

import urllib

req = urllib.urlopen("http://twitter.com/followers/ids/44446416.xml?page=1")
content = req.read()

print content
```

Figura 2.12. Exemplo do uso da biblioteca URLLIB em Python

Existem várias ferramentas que podem ser utilizadas para se coletar dados de redes sociais online. Como cada pesquisa requer um tipo de coleta e cada coleta de dados possui sua particularidade, desenvolver o próprio coletor pode ser necessário. A figura 2.11 mostra o uso da biblioteca LWP na linguagem Perl. Este código realiza a coleta dos seguidores de um usuário no Twitter através de sua API. De maneira similar, o código em Python da figura 2.12 utiliza a biblioteca URLLIB para realizar a mesma tarefa. O resultado da execução dos crawlers é a lista de seguidores de um usuário do Twitter em formato XML, como ilustra a figura 2.13.

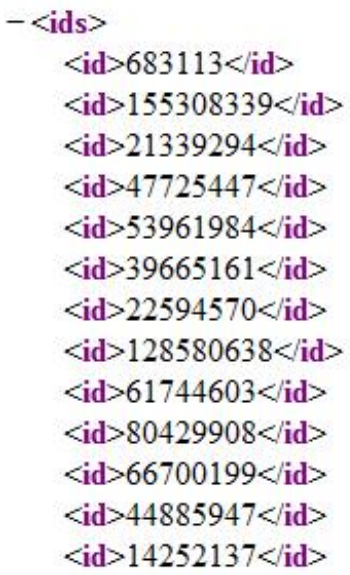


Figura 2.13. Resultado da execução dos crawlers em Perl e Python

2.4.3.6. Ética dos crawlers

Coletar dados da Web não é apenas útil para a pesquisa sobre redes sociais, mas constitui um importante passo para qualquer mecanismo de busca, como o Google [35]. Entretanto, crawlers ou robôs são ferramentas com habilidade para causar vários problemas para empresas relativos à coleta e disponibilização de conteúdo indevido. Sendo assim, foi convencionado através de um protocolo conhecido como robots.txt que sítios Web podem regulamentar o que pode e o que não pode ser coletado no sistema. Este método é bastante utilizado pelos administradores de sistemas para informar aos robôs visitantes quais diretórios de um sítio Web não devem ser coletados. O robots.txt nada mais é do que um arquivo que fica no diretório raiz dos sítios e contém regras para robôs específicos ou de uso geral para qualquer robô. Ao visitar um site, os robôs devem buscar primeiro pelo arquivo robots.txt e verificam suas permissões. Exemplos desses arquivos seriam:

http://portal.acm.org/robots.txt
http://www.google.com/robots.txt

http://www.globo.com/robots.txt
http://www.orkut.com.br/robots.txt
http://www.youtube.com/robots.txt
http://www.robotstxt.org/robots.txt

A seguir mostramos um exemplo simples de regra em um arquivo robots.txt. Essa regra restringe todos os crawlers de acessarem qualquer conteúdo no sistema.

```
User-agent: *
Disallow: /
```

É possível ainda especificar restrições a alguns robôs específicos ou restringir o acesso a alguns diretórios específicos. Como exemplo, o sítio Web www.globo.com oferece restrições para todos os robôs nos seguintes diretórios.

```
User-agent: *

Disallow: /PPZ/
Disallow: /Portal/
Disallow: /Java/
Disallow: /Servlets/
Disallow: /GMC/foto/
Disallow: /FotoShow/
Disallow: /Esportes/foto/
Disallow: /Gente/foto/
Disallow: /Entretenimento/Ego/foto/
```

No caso de coleta de redes sociais é importante verificar não só o arquivo robots.txt dos sistemas, mas também os termos de uso do sistema.

2.4.4. Dados de aplicações

Em uma tentativa bem sucedida de enriquecer a experiência dos usuários de redes sociais online, o Facebook realizou uma de suas maiores inovações: abriram sua plataforma para desenvolvedores de aplicações [4]. Com esta inovação, desenvolvedores são capazes de criar diferentes tipos de aplicações. Com o sucesso no Facebook, outros sistemas como Orkut e MySpace também adotoram essa estratégia. O tipo e número de aplicações nesses sistemas se tornou ilimitada com o novo modelo de API aberta implementado. O Facebook sozinho possui mais de 81,000 aplicações [2]. Empresas como a Zynga, especializadas em desenvolver essas aplicações, contam com mais de 80 milhões de usuários registrados em suas aplicações [2]. Uma grande lista de aplicações do Facebook pode ser encontrada na seguinte referência [3].

Uma estratégia para se obter dados de redes sociais online é através do desenvolvimento de aplicações sociais. A figura 2.14 mostra o funcionamento de uma aplicação terceirizada em execução em redes sociais online como o Facebook ou o Orkut. Aplicações são caracterizadas pela presença de um servidor da rede social intermediando toda

a comunicação entre o cliente e o servidor da aplicação. Tipicamente, o cliente envia a requisição ao servidor da rede social online, que repassa ao servidor da aplicação. Então, o servidor da aplicação manda de volta a resposta ao servidor da rede social, que repassa ao cliente [73].

Aplicações podem ser utilizadas para o estudo de interações entre os usuários que utilizam as aplicações e também podem ser úteis para coletar outras informações dos usuários. Como exemplo, aplicações podem pedir permissão aos usuários para acessar informações como lista de amigos e atividades executadas dentro de uma sessão.

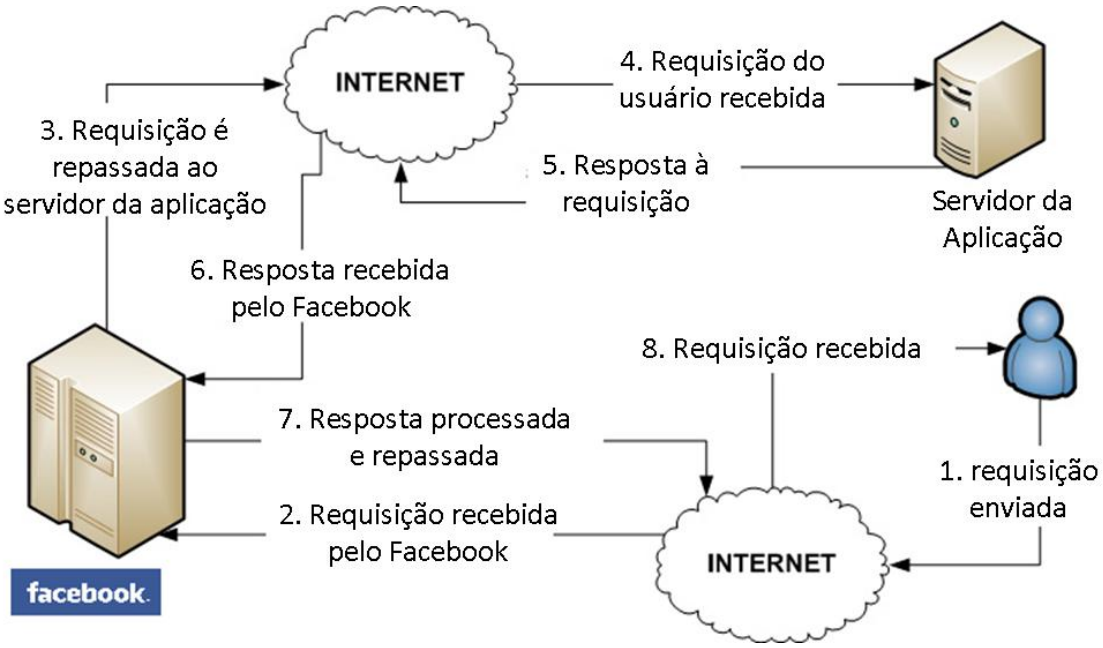


Figura 2.14. Funcionamento de aplicações no Facebook e no Orkut

Alguns trabalhos fizeram uso dessa estratégia de medição para estudar os usuários de redes sociais online. Nazir e colaboradores [72] analisaram características de aplicações no Facebook, desenvolvendo e lançando suas próprias aplicações. Em particular, eles estudaram a formação de comunidades online a partir de grafos de interação entre os usuários de suas aplicações. Mais recentemente, Nazir e colaboradores [73] estudaram várias características relacionadas ao desempenho de suas aplicações no Facebook.

2.5. Conclusões

Redes sociais se tornaram extremamente populares e parte do nosso dia a dia, causando o surgimento de uma nova onda de aplicações disponíveis na Web. A cada dia, grandes quantidades de conteúdo são compartilhadas e milhões de usuários interagem através de elos sociais. Apesar de tanta popularidade, o estudo de redes sociais ainda está em sua infância, já que estes ambientes estão ainda experimentando novas tendências e enfrentando diversos novos problemas e desafios.

Redes sociais compõem ambientes perfeitos para o estudo de vários temas da computação, incluindo sistemas multimídia e interação humano-computador. Além disso, por permitir que usuários criem conteúdo, redes sociais vêm se tornando um tema chave

em pesquisas relacionadas à organização e tratamento de grandes quantidades de dados, além de constituírem um ambiente ideal para extração de conhecimento e aplicação de técnicas de mineração de dados.

Este trabalho oferece uma introdução ao pesquisador que pretende explorar o tema. Inicialmente, foram apresentadas as principais características das redes sociais mais populares atualmente. Em seguida, discutimos as principais métricas e tipos de análises utilizadas no estudo dos grafos que formam a topologia das redes sociais. Finalmente, sumarizamos as principais abordagens utilizadas para se obter dados de redes sociais online e discutimos trabalhos recentes que utilizaram essas técnicas.

Agradecimentos

Este trabalho foi parcialmente financiado pelo Instituto Nacional de Ciência e Tecnologia para a Web.

Referências

[1] comscore: Americans viewed 12 billion videos online in may 2008. http://www.comscore.com/press/release.asp?press=2324. Acessado em Março/2010.

[2] Developer analytics. http://www.developeranalytics.com. Acessado em Março/2010.

[3] Facebook application directory. http://www.facebook.com/apps. Acessado em Março/2010.

[4] Facebook platform. http://developers.facebook.com. Acessado em Março/2010.

[5] Facebook Press Room, Statistics. http://www.facebook.com/press/info.php?statistics. Acessado em Março/2010.

[6] Gnuplot. http://www.gnuplot.info/. Acessado em Agosto/2010.

[7] Google OpenSocial. http://code.google.com/apis/opensocial/. Acessado em Março/2010.

[8] List of social network web sites. http://en.wikipedia.org/wiki/List_of_social_networking_websites. Acessado em Março/2010.

[9] Matlab. http://www.mathworks.com/products/matlab/. Acessado em Agosto/2010.

[10] Needle in a Haystack: Efficient Storage of Billions of Photos. Facebook Engineering Notes, http://tinyurl.com/cju2og. Acessado em Março/2010.

[11] New york times. a web site born in u.s. finds fans in brazil. http://www.nytimes.com/2006/04/10/technology/10orkut.html. Acessado em Março/2010.

[12] New york times. uploading the avantgarde. http://www.nytimes.com/2009/09/06/magazine/06FOB-medium-t.htm. Acessado em Julho/2010.

[13] Yahoo! pipes. http://pipes.yahoo.com/pipes. Acessado em Agosto/2010.

[14] YouTube fact sheet. http://www.youtube.com/t/fact_sheet. Acessado em Março/2010.

[15] L. Adamic, O. Buyukkokten, and E. Adar. A social network caught in the web. *First Monday*, 8(6), 2003.

[16] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. In *World Wide Web Conference (WWW)*, pages 835–844, 2007.

[17] R. Albert, H., Jeong, and A. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000.

[18] R. Albert, H. Jeong, and A. Barabasi. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

[19] N. Ali-Hasan and L. Adamic. Expressing social relationships on the blog through links and comments. In *AAAI Conference on Weblogs and Social Media (ICWSM)*, 2007.

[20] A. Amaral, A. Scala, M. Barthelemy, and E. Stanley. Classes of small-world networks. 97(21):11149–11152, 2000.

[21] B. Williamson. Social network marketing: ad spending and usage. *EMarketer Report*, 2007. http://tinyurl.com/2449xx. Acessado em Março/2010.

[22] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439), 1999.

[23] F. Benevenuto, F. Duarte, T. Rodrigues, V. Almeida, J. Almeida, and K. Ross. Understanding video interactions in YouTube. In *ACM Conference on Multimedia (MM)*, pages 761–764, 2008.

[24] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting spammers on twitter. In *7th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, 2010.

[25] F. Benevenuto, A. Pereira, T. Rodrigues, V. Almeida, J. Almeida, and M. Gonçalves. Avaliação do perfil de acesso e navegação de usuários em ambientes web de compartilhamento de vídeos. In *Brazilian Symposium on Multimedia Systems and Web (WebMedia)*, pages 149–156, 2009.

[26] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, and M. Gonçalves. Detecting spammers and content promoters in online video social networks. In *Int'l ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 620–627, 2009.

[27] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, M. Gonçalves, and K. Ross. Video pollution on the web. *First Monday*, 15(4), April 2010.

[28] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, and K. Ross. Video interactions in online video social networks. *ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP)*, 5(4):1–25, 2009.

[29] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, C. Zhang, and K. Ross. Identifying video spammers in online social networks. In *Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 45–52, 2008.

[30] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing user behavior in online social networks. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 49–62, 2009.

[31] J. Binder, A. Howes, and A. Sutcliffe. The problem of conflicting social spheres: effects of network structure on experienced tension in social network sites. In *ACM SIGCHI Conference on Human factors in Computing Systems (CHI)*, pages 965–974, 2009.

[32] D. Boyd. *Why Youth (Heart) Social Network Sites: The Role of Networked Publics in Teenage Social Life*. Cambridge, MA, 2007.

[33] D. Boyd and N. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1-2), 2007.

[34] V. Braitenberg and A. Schüz. *Cortex: Statistics and Geometry of Neuronal Connectivity*. Springer-Verlag, 1998.

[35] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[36] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks*, 33:309–320, 2000.

[37] M. Burke, C. Marlow, and T. Lento. Feed me: Motivating newcomer contribution in social network sites. In *ACM SIGCHI Conference on Human factors in Computing Systems (CHI)*, pages 945–954, 2009.

[38] M. Cha, H. Haddadi, F. Benevenuto, and K. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *In 4th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2010.

[39] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 1–14, 2007.

[40] M. Cha, A. Mislove, and K. Gummadi. A measurement-driven analysis of information propagation in the Flickr social network. In *World Wide Web Conference (WWW)*, pages 721–730, 2009.

[41] C. Chapman and M. Lahav. International ethnographic observation of social networking sites. In *ACM SIGCHI Conference on Human factors in Computing Systems (CHI)*, pages 3123–3128, 2008.

[42] P. Chatterjee, D. L. Hoffman, and T. P. Novak. Modeling the clickstream: implications for web-based advertising efforts. *Marketing Science*, 22(4):520–541, 2003.

[43] D. Chau, Pandit, S. Wang, and C. Faloutsos. Parallel crawling for online social networks. In *World Wide Web Conference (WWW)*, pages 1283–1284, 2007.

[44] H. Chun, H. Kwak, Y. Eom, Y.-Y. Ahn, S. Moon, and H. Jeong. Comparison of online social relations in volume vs interaction: a case study of Cyworld. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 57–70, 2008.

[45] G. Cormode and B. Krishnamurthy. Key differences between web 1.0 and web 2.0. *First Monday*, 13(6), 2008.

[46] X. Dale and C. Liu. Statistics and social network of YouTube videos. In *Int'l Workshop on Quality of Service (IWQoS)*, 2008.

[47] F. Duarte, F. Benevenuto, V. Almeida, and J. Almeida. Locality of reference in an hierarchy of web caches. In *IFIP Networking Conference (Networking)*, pages 344–354, 2006.

[48] F. Duarte, B. Mattos, A. Bestavros, V. Almeida, and J. Almeida. Traffic characteristics and communication patterns in blogosphere. In *Conference on Weblogs and Social Media (ICWSM)*, 2007.

[49] H. Ebel, L. Mielsch, and S. Bornholdt. Scale free topology of e-mail networks. *Physical Review E*, 66(3):35103, 2002.

[50] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 251–262, 1999.

[51] E. Gabrilovich, S. Dumais, and E. Horvitz. Newsjunkie: Providing personalized newsfeeds via analysis of information novelty. In *World Wide Web Conference (WWW)*, pages 482–490, 2004.

[52] D. Garlaschelli and M. Loffredo. Patterns of link reciprocity in directed networks. *Physical Review Letters*, 93(26):268701, 2004.

[53] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. YouTube traffic characterization: A view from the edge. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 15–28, 2007.

[54] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Characterizing user sessions on YouTube. In *IEEE Multimedia Computing and Networking (MMCN)*, 2008.

[55] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[56] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Int'l. Conference on Very Large Data Bases (VLDB)*, pages 576–587, 2004.

[57] A. Joinson. Looking at, looking up or keeping up with people?: motives and use of Facebook. In *ACM SIGCHI Conference on Human factors in Computing Systems (CHI)*, pages 1027–1036, 2008.

[58] R. King. When your social sites need networking, *BusinessWeek*, 2007. `http://tinyurl.com/o4myvu`. Acessado em Março/2010.

[59] B. Krishnamurthy. A measure of online social networks. In *Conference on Communication Systems and Networks (COMSNETS)*, 2009.

[60] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

[61] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Int'l World Wide Web Conference (WWW)*, 2010.

[62] S. Lee, P. Kim, and H. Jeong. Statistical properties of sampled networks. *Physical Review E*, 73(30):102–109, 2006.

[63] K. Lerman. Social information processing in news aggregation. *IEEE Internet Computing*, 11(6):16–28, 2007.

[64] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):228–237, 2007.

[65] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *World Wide Web Conference (WWW)*, 2008.

[66] L. Li, D. Alderson, J. Doyle, and W. Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4), 2005.

[67] A. Mahanti, D. Eager, and C. Williamson. Temporal locality and its impact on web proxy cache performance. *Performance Evaluation Journal*, 42(2-3):187–203, 2000.

[68] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, May 1967.

[69] A. Mislove. *Online Social Networks: Measurement, Analysis, and Applications to Distributed Information Systems*. PhD thesis, Rice University, Department of Computer Science, 2009.

[70] A. Mislove, M. Marcon, K. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 29–42, 2007.

[71] C. Moore and M. Newman. Epidemics and percolation in small-world networks. *Physical Review E*, 61(5):5678, 2000.

[72] A. Nazir, S. Raza, and C. Chuah. Unveiling facebook: A measurement study of social network based applications. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 43–56, 2008.

[73] A. Nazir, S. Raza, D. Gupta, C. Chua, and B. Krishnamurthy. Network level footprints of facebook applications. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 63–75, 2009.

[74] M. Newman. The structure of scientific collaboration networks. 98(2):404–409, 2001.

[75] M. Newman. Assortative mixing in networks. *Physical Review E*, 89(20):208701, 2002.

[76] M. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[77] M. Newman. Coauthorship networks and patterns of scientific collaboration. 101(1):5200–5205, 2004.

[78] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):26113, 2004.

[79] J. Otterbacher. 'helpfulness' in online communities: a measure of message quality. In *ACM SIGCHI Conference on Human factors in Computing Systems (CHI)*, pages 955–964, 2009.

[80] N. O. Report. Social networks & blogs now 4th most popular online activity, 2009. `http://tinyurl.com/cfzjlt`. Acessado em Março/2010.

[81] P. Rodriguez. Web infrastructure for the 21st century. *WWW'09 Keynote, 2009*. `http://tinyurl.com/mmmaa7`. Acessado em Março/2010.

[82] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger. Understanding online social network usage from a network perspective. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 35–48, 2009.

[83] S. Schroeder. 20 ways to aggregate your social networking profiles, *Mashable*, 2007. `http://tinyurl.com/2ceus4`. Acessado em Março/2010.

[84] J. Thom-Santelli, M. Muller, and D. Millen. Social tagging roles: publishers, evangelists, leaders. In *ACM SIGCHI Conference on Human factors in Computing Systems (CHI)*, pages 1041–1044, 2008.

[85] M. Torkjazi, R. Rejaie, and W. Willinger. Hot today, gone tomorrow: On the migration of myspace users. In *ACM SIGCOMM Workshop on Online social networks (WOSN)*, pages 43–48, 2009.

[86] K. S. Trivedi. *Probability and statistics with reliability, queuing and computer science applications*. John Wiley and Sons Ltd., Chichester, UK, 2002.

[87] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in Facebook. In *ACM SIGCOMM Workshop on Online Social Networks (WOSN)*, pages 37–42, 2009.

[88] S. Wasserman, K. Faust, and D. Iacobucci. *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, 1994.

[89] D. Watts. *Small Worlds: the Dynamics of Networks Between Order and Randomness*. Princeton University Press, 1999.

[90] D. Watts. A simple model of global cascades on random networks. 99(9):5766–5771, 2002.

[91] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *ACM international conference on Web search and data mining (WSDM)*, pages 261–270, 2010.

[92] C. Williamson. On filter effects in web caching hierarchies. *ACM Transactions on Internet Technology (TOIT)*, 2(1):47–77, 2002.

[93] J. Zhang, M. Ackerman, and L. Adamic. Expertise networks in online communities: Structure and algorithms. In *World Wide Web Conference (WWW)*, pages 221–230, 2007.

[94] M. Zink, K. Suh, Y. Gu, and J. Kurose. Watch global, cache local: YouTube network traces at a campus network - measurements and implications. In *IEEE Multimedia Computing and Networking (MMCN)*, 2008.

Capítulo 3

Desenvolvimento de Aplicações Imperativas para TV Digital no *middleware* Ginga com Java

Raoni Kulesza[1], Carlos Alberto Saibel Santos[2], Tatiana Aires Tavares[1], Manoel Marques Neto[2], Guido Lemos de Souza Filho[1]

[1] Laboratório de Aplicações de Vídeo Digital (LAVID), Departamento de Informática, UFPB
{raoni, tati, guido}@lavid.ufpb.br

[2] Doutorado Multiinsitucioal em Ciência da Computação
Departmento de Ciência da Computação, IM, UFBA
saibel@ufba.br; manoelnetom@gmail.com

Abstract

This chapter presents an introduction to Java applications developing for interactive digital TV based on Ginga, middleware standard of Brazilian System of Digital TV. Java applications in Ginga use a set of APIs based on the JavaDTV specification and brazilian innovations, recently recognized internationally. The chapter presented the features of these solutions through the sample application construction step by step. The chapter also presents techniques and tools to support software engineering to facilitate the development and contribute to the reuse of application code.

Resumo

Este capítulo apresenta uma introdução ao desenvolvimento de aplicações Java para TV digital interativa com base no middleware Ginga, padrão do Sistema Brasileiro de TV Digital. No Ginga as aplicações Java utilizam um conjunto de APIs baseado na especificação JavaDTV e inovações brasileiras, reconhecidas recentemente em âmbito internacional. O capítulo apresentada as funcionalidades dessas soluções através da construção passo a passo de uma aplicação. O capítulo também apresenta técnicas e ferramentas de apoio a engenharia de software que visam facilitar o desenvolvimento e contribuir para o reuso do código das aplicações geradas.

3.1. Introdução

A possibilidade de veiculação de softwares e dados juntamente com os fluxos de áudio e vídeo transmitidos pelas emissoras é uma das maiores inovações trazidas pelos sistemas de TV digital. Esta inovação cria oportunidades para a produção e desenvolvimento de serviços e aplicações dos mais diversos tipos, voltados para esta nova mídia interativa.

Este capítulo explora os desafios e novas oportunidades para o profissional de software advindos da implantação do Sistema Brasileiro de TV Digital (SBTVD ou ISDB-Tb) terrestre[1]. O objetivo do capítulo é atualizar o profissional de software com relação aos novos modelos de produção de conteúdo para a TV digital integrando software e conteúdo audiovisual. Para tanto, primeiramente, a definição do conceito de TV interativa é analisado tendo como base uma série de referências da literatura. Na seção seguinte, discute-se o aparecimento da TV digital no contexto de um processo mais amplo de convergência digital, caracterizado pela fusão de tecnologias e facilidade de produção e acesso de conteúdo. Em seguida, as aplicações interativas são discutidas, procurando caracterizar a parte como o projeto da interatividade impacta na geração desse novo formato de conteúdo televisivo. Chega-se então ao processo de desenvolvimento dos programas interativos e de como estes novos requisitos são importantes para o desenvolvimento de software para a TV. Por fim, a plataforma disponível para o desenvolvimento de aplicações interativas para TV Digital no Brasil é abordada. Em especial, a parte do *middleware* Ginga destinada a execução de programas em Java, denominada Ginga-J, é detalhada e um exemplo de aplicação é apresentado.

3.2. Interatividade na TV

Pode parecer estranho, mas o conceito de TV Interativa (TVi) já existe há mais de 50 anos[2]. Por outro lado, a definição precisa do que é chamado de TVi ainda é uma questão que está longe de ser resolvida, conforme mostra a Tabela 3.1. Mais ainda, termos como TV Digital Interativa (TVDI), Web TV, TV na Internet e outros são muitas vezes utilizados como sinônimos de TVi[3].

A inclusão da interatividade nos programas de TV tornou-se um grande atrativo para as indústrias de radiodifusão e para os geradores de conteúdo que estão sempre em busca de novas maneiras de fidelizar seus telespectadores. Alguns exemplos recentes de programas líderes de audiência utilizam a colaboração (interação) do telespectador para: (1) por meio de mecanismos de preferência popular, definir os rumos do programa (Big Brother Brasil); (2) produzir e enviar conteúdos a serem integrados aos programas (Quadro Bola Cheia Bola Murcha do Fantástico); ou ainda (3) interagir com participantes de um programa (mensagens com questões para os comentaristas durante uma partida de futebol).

---

[1] O ISDB-Tb, baseado no padrão japonês ISDB-T, tinha sido adotado por Peru, Argentina, Chile, Venezuela, Equador, Costa Rica, Paraguai, Filipinas e Bolívia no momento da escrita deste texto.

[2] O programa "Winky Dink And You", produzido pela CBS-TV na década de 50, é considerado por muitos como o primeiro programa interativo para a TV http://www.toontracker.com/winky/winky.htm

[3] Em http://www.itvdictionary.com/itv.html dezenas de termos associados à TV interativa são listados.

Este novo desenho de conteúdo televisivo tem impactos diretos na forma com a qual se assiste a esse conteúdo (recepção) como também na forma com a qual este conteúdo é feito (produção). O processo de produção de conteúdo para TV passa a englobar o desenvolvimento de software (interatividade), e, portanto, passa a ser caracterizado como um novo processo, não linear, iterativo, ágil, multidisciplinar e convergente.

Tabela 3.1: Definições de TV Interativa encontradas na literatura

Mark Gawlinski [Gawlinski 2003, p. 5]	[...] algo que permite que o telespectador ou telespectadores e as pessoas que fazem um canal de televisão, programa ou serviço se engajem em um diálogo. Mais especificamente, pode ser definida como um diálogo que leva os telespectadores a além da experiência passiva de assistir e os permita fazer escolhas e tomar ações.
Konstantinos Chorianopoulos [Chorianopoulos 2004, p. 9]	[…] é um termo genérico utilizado para todos os sistemas de televisão que oferecem ao consumidor interatividade além da troca de canais e do teletexto.
Jerry Whitaker [Whitaker 2001]	[...] tudo o que permite que um consumidor interaja com o sistema usando um controle remoto ou um teclado para acessar novos e avançados serviços.
Robert Schwalb [Schwalb 2003]	[...] é a coleção de serviços que suporta escolhas e ações iniciadas pelo telespectador e que são relacionadas a um ou mais canais de programação de vídeo.
Karyn Y. Lu [Lu 2005]	[...] qualquer programa de televisão que incorpora conteúdo adicional ou algum tipo de interatividade com o usuário. [...] termo genérico que cobre a convergência da televisão com tecnologias das mídias digitais como computadores, PVRs, jogos eletrônicos, dispositivos móveis e sem fio, possibilitando a interação com o usuário.

A intenção deste capítulo não é apresentar uma definição única e definitiva para o termo TVi, mas apresentar a possibilidade de construir aplicações interativas usando a linguagem de programação Java. Mais especificamente, o capítulo trata de um caso específico de aplicações: aquelas **enviadas aos receptores**[4] **por uma emissora de TV juntamente com seus programas, utilizando o canal que está sob seu controle**.

[4] Os termos receptor, receptor de TV, set-top-box ou terminal de acesso são utilizados indistintamente na sequência do texto.

3.3. Convergência Digital

O processo de convergência digital pode ser encarado sobre dois diferentes pontos de vista. No primeiro, a convergência é vista como um casamento de tecnologias ou indústrias que podem se tornar: (1) competitivas ou (2) complementares (ou, melhor ainda, dependentes). No segundo, ele pode ser visto como uma (re)união de diferentes tipos de mídia por meio de uma tecnologia única. Um exemplo de competitividade criada pela convergência é a do acesso a conteúdos de jornais e outras mídias de comunicação pela Web, que resultou em profundas mudanças nas nos jornais e agências de notícias, na indústria fonográfica entre outras. Recentemente, em julho de 2010, o Jornal do Brasil, que já foi um dos mais importantes jornais da imprensa brasileira, além de ser o primeiro jornal do País a ter versão Web, anunciou o fim da sua versão impressa. Os canais de TV e os seus provedores de vídeo na Web são os exemplos mais recentes de formas de acesso a conteúdo audiovisual que competem com o modelo tradicional de comunicação de massa da TV *broadcast*.

O processo de convergência, além dos ganhos tecnológicos, possui também uma forte vertente sócio-econômica que implica na adoção e uso coordenado de protocolos e padrões comuns pelas partes que se integram. Em outras palavras, produtores e consumidores devem adotar plataformas compatíveis, além de utilizarem padrões específicos (para codificação, transmissão e armazenagem) para permitir o trabalho em conjunto no mundo convergente.

A TV digital, que integra a capacidade de processamento aos receptores de TV, também é um dos produtos gerados neste processo de convergência. Mais ainda, a TV digital surge em um momento no qual o modelo de distribuição de conteúdo por *broadcast* tem que fazer face aos novos modelos de produção e distribuição compartilhada e sob demanda de conteúdos audiovisuais pela Web. Neste cenário, o desafio para a produção de conteúdo audiovisual está na criação de novas aplicações e de formas de distribuição de conteúdo que integrem as características de qualidade visual, simplicidade de acesso, confiabilidade de transmissão e baixo custo, oriundos do modelo da TV tradicional, com o acesso personalizado, busca de informações sob demanda, compartilhamento de conteúdo e comunicação entre grupos do modelo Web. O uso de TVs com acesso direto à Internet (as chamadas TV conectadas) e o desenvolvimento de aplicações do tipo *widgets* (moderadas pelas emissoras, fabricantes ou algum provedor) para acesso a recursos Web não solucionam a questão, por outro lado acirram a competitividade entre os dois mundos. Entretanto, pensar na TV digital como uma plataforma integrada de acesso a serviços, conteúdos e, principalmente, a pessoas, parece ser um caminho mais promissor. A chamada TV Social (apontada pelo *MIT Technology Review* como uma das 10 mais importantes tecnologias emergentes para os próximos anos) aparece como um caminho interessante para o problema, onde a convergência tecnológica transforma a TV num dispositivo central para a integração de indivíduos com interesses semelhantes [Cesar 2009].

A evidente melhoria da qualidade do conteúdo visual com a digitalização do sistema de TV, novos problemas devem ser enfrentadas pelas emissoras na geração de conteúdo interativo para TV digital. No modelo de *broadcast* tradicional, toda a produção do conteúdo fica sob responsabilidade da emissora, a qual integra uma série de elementos (ícones, vinhetas, animações) sobre um conteúdo único audiovisual, que é

transmitido sobre o canal controlado pela emissora. Nesse modelo, o receptor de TV sintonizado no canal da emissora decodifica o sinal e apresenta o conteúdo integrado produzido a uma certa taxa de quadros e de forma sincronizada com o áudio correspondente. Quando conteúdos interativos são enviados pelas emissoras, porções (no tempo e no espaço) do conteúdo integrado podem gerar algum tipo de processamento (surgimento de informações extras na tela, disparo de um elo, mudança de ângulo da câmera) ao serem acessadas pelo usuário-telespectador através do seu controle remoto. Para que esse processamento seja possível, uma série de requisitos deve ser satisfeitos, tanto da parte de quem gera, de quem transmite, quanto de quem recebe e processa essas aplicações.

Para que todas as partes da cadeia de produção, distribuição e consumo de conteúdos audiovisuais interativos para TV digital se encaixem perfeitamente, é necessário que essas partes utilizem padrões comuns. Nesse sentido, o SBTVD especifica vários padrões de referência, dentre os quais se destacam o H.264 para a codificação de vídeo, o MPEG-4 para o áudio, além do MPEG-2 System para o transporte (multiplexação) dos fluxos de áudio, vídeo e dados. Dentre os padrões estabelecidos, porém, a camada *middleware* Ginga é, sem dúvida, a mais importante diferença do SBTVD com relação aos outros padrões internacionais.

Uma das funções do *middleware* Ginga é dar suporte ao desenvolvimento de aplicações para a plataforma de TV digital associada ao SBTVD. De forma semelhante à abordagem adotada por outros sistemas de TV digital, o Ginga tem uma estrutura composta por dois subsistemas: (1) o de apresentação, que dá suporte a execução de aplicações declarativas NCL [Soares 2007] e (2) o de execução, que dá suporte a aplicações imperativas Java [Souza Filho 2007], que são o foco deste capítulo.

3.4. Aplicações para TV Digital

A partir do momento em que os receptores de TV foram dotados de poder de processamento e que dados e objetos puderam ser transmitidos junto ao conteúdo audiovisual dos programas, criou-se um novo mercado para os desenvolvedores de software. Esses softwares permitem agregar uma série de funcionalidades e serviços à cadeia de produção de conteúdo para a plataforma de TV digital.

3.4.1 Paradigmas Imperativo e Declarativo

O tipo de funcionalidade a ser adicionada a um programa de TV digital influencia diretamente o tipo de linguagem (paradigma) a ser utilizado na implementação. Assim como em outros domínios, (como na Web) as aplicações para TV digital são, normalmente, construídas usando abordagens linguagens declarativas, imperativas (algumas vezes também chamadas de procedurais) ou ainda, com o uso de uma abordagem híbrida, integrando os dois tipos de linguagens.

As linguagens declarativas tendem a ser mais intuitivas, à primeira vista e, portanto, mais simples por programadores (também chamados autores, nesse caso), os quais definem: (1) restrições espaciais e temporais (sincronismo) entre as mídias que compõem a aplicação interativa e (2) tratamento de eventos de interação do usuário com aplicação. Uma linguagem declarativa enfatiza a descrição declarativa do problema, ao invés da sua decomposição em uma implementação algorítmica. Elas são linguagens de

mais alto nível de abstração, usualmente ligadas a um domínio ou objetivo específico, onde o programador fornece apenas o conjunto das tarefas a serem realizadas, não estando preocupado com os detalhes de como elas serão executadas [Soares 2007]. A linguagem declarativa utilizada no contexto do SBTVD é a NCL.

As linguagens imperativas tendem a ser mais apropriadas para aplicações genéricas, orientadas a eventos, nas quais o programador possui um maior controle do código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa. Neste caso, processador do receptor deve ser informado sobre cada passo a ser executado. Pode-se afirmar que, em linguagens imperativas, o programador possui um maior poder sobre o código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa. A linguagem mais usual encontrada nos ambientes imperativos dos sistemas de TV digital é a Java. Uma aplicação para TV desenvolvida nesta linguagem é também conhecida como Xlet.

As aplicações imperativas podem referenciar conteúdos declarativos, ou até construir e iniciar a apresentação de um conteúdo declarativo. Estas aplicações são denominadas de híbridas constituem uma poderosa ferramenta de produção de conteúdo para TV digital, ao unir as vantagens dos paradigmas declarativo e imperativo. Como os dois tipos de aplicações devem coexistir no receptor, todos os sistemas de TV digital propostos no mundo trabalham com a idéia de que o receptor possui um *middleware* que dê suporte a ambos os paradigmas.

3.4.2 Processando Dados no Receptor de TV

As funcionalidades de um programa de TV dependem, dentre outros fatores, dos dados e objetos que irão ser processados pelas aplicações. A depender da sua natureza, os dados e objetos podem ser: (1) enviados pela própria emissora de TV, em broadcast, junto ao conteúdo audiovisual; (2) acessados por meio do canal de interatividade ou (3) acessados, de forma alternativa, a partir de um repositório (através de uma porta USB, por exemplo). A grade de programação de uma emissora (EPG ou *Eletronic Programming Guide*), por não conter dados pessoais do usuário e não ser composta por arquivos extensos é um exemplo típico de aplicação que pode ser enviada junto com os programas de uma emissora. A visualização de mensagens de e-mail, devido ao seu caráter pessoal, seria um exemplo de aplicação com dados enviados por intermédio do canal de interatividade. Outro ponto a ser observado é o fato do canal de interatividade possibilitar que o usuário receba apenas as informações que solicita, enquanto que os dados recebidos pelo canal controlado pela emissora são enviados mesmo sem solicitação do usuário.

A oferta de serviços é dependente da infraestrutura disponível ao usuário. Esta estrutura é formada pelo decodificador, pela geradora de conteúdo e pelo provedor de serviços (no caso de dependência de um meio de comunicação, ou canal de interatividade, para que os usuários acessem os serviços interativos). Os novos serviços incluem os tradicionais de previsão do tempo ou cotação de ações da bolsa, além de acesso a redes sociais, navegação Web, entre outros.

A transmissão de dados e objetos gerados pelas emissoras e processados nos receptores de TV é feita de acordo com um mecanismo denominado carrossel

3.4.3 Carrossel de Dados

O carrossel de dados e/ou de objetos é o mecanismo responsável por enviar de forma cíclica programas e dados multiplexados ao conteúdo audiovisual dos programas no fluxo de transporte MPEG-2. O carrossel funciona como um disco virtual que armazenados e aplicativos a serem disponibilizados aos usuários. Todos os sistemas de TV digital terrestre utilizam o protocolo especificado no padrão DSM-CC (*Digital Storage Media – Command and Control*) [ISO/IEC 13818-6 1998]. Através do carrossel de objetos torna-se possível remontar no receptor a estrutura de diretórios da aplicação e seus dados da maneira em que se encontravam no gerador de conteúdo. Com o envio cíclico de conteúdo por meio do carrossel é possível que programas e dados sejam transmitidos corretamente para o receptor mesmo se o canal for sintonizado após o início da transmissão do programa interativo ou que partes dos programas ou dados sejam perdidas. Para aplicações que utilizam dados e aplicações provenientes do canal de interatividade (por exemplo, acessar uma aplicação de correio eletrônico pela TV) ou dados locais (por exemplo, a apresentação de fotos armazenadas em um *pendrive* conectado à porta USB do receptor), não é necessário o acesso à estrutura do carrossel.

3.4.4. Concepção de Aplicações Interativas

O software transmitido juntamente com o conteúdo audiovisual de um programa de TV[5] é chamado, na TV Interativa, de aplicativo ou aplicação. No entanto, as aplicações interativas têm algumas particularidades em relação às tradicionais aplicações de software [Veiga 2007]:

1. Elas podem fazer parte de um programa de TV, o qual tem formato e contexto próprios;
2. Apesar de serem oferecidas por meio de um *front-end* único (a TV), podem atender a mais de um usuário. De fato, a TV atende, de forma coletiva, a um grupo de usuários, que possuem perfis particulares e cujos anseios em um veículo de comunicação de massa são mapeados em diversidade na grade de programação;
3. Elas devem lidar com interatividade (em diferentes níveis) e uma possível organização não-linear de conteúdo.

O tipo de funcionalidade a ser adicionada a um programa de TV digital influencia diretamente o tipo de linguagem (paradigma) a ser utilizado na implementação. Assim como em outros domínios, (como na Web) as aplicações para TV digital estão, normalmente, associadas aos paradigmas declarativo, imperativo (algumas vezes também chamado de procedural) ou híbrido.

As linguagens declarativas tendem a ser mais intuitivas, à primeira vista e, portanto, mais simples por programadores (também chamados autores, nesse caso), os quais definem: (1) restrições espaciais e temporais (sincronismo) entre as mídias que compõem a aplicação interativa e (2) tratamento de eventos de interação do usuário com aplicação. Uma linguagem declarativa enfatiza a descrição declarativa do problema, ao invés da sua decomposição em uma implementação algorítmica. Elas são linguagens de

[5]Refere-se a um programa de TV Convencional, doravante denominado apenas de programa de TV.

mais alto nível de abstração, usualmente ligadas a um domínio ou objetivo específico, onde o programador fornece apenas o conjunto das tarefas a serem realizadas, não estando preocupado com os detalhes de como elas serão executadas [Soares 2007]. A linguagem declarativa utilizada no contexto do SBTVD é a NCL.

As linguagens imperativas tendem a ser mais apropriadas para aplicações genéricas, orientadas a eventos, nas quais o programador possui um maior controle do código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa. Como os dois tipos de aplicações devem coexistir num ambiente de TV digital, todos os sistemas proposto trabalham com a idéia de que o receptor possui um *middleware* que dê suporte a ambos os paradigmas. Neste cenário, o computador deve ser informado sobre cada passo a ser executado. Pode-se afirmar que, em linguagens procedurais, o programador possui um maior poder sobre o código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa. Entretanto, para isso, ele deve ser bem qualificado e conhecer bem os recursos de implementação. A linguagem mais usual encontrada nos ambientes imperativos de um sistema de TV digital é a Java. Uma aplicação para TV digital desenvolvida nesta linguagem é também conhecida como *Xlet*. As aplicações imperativas podem referenciar conteúdos declarativos, ou até construir e iniciar a apresentação de um conteúdo declarativo. Estas aplicações são denominadas de híbridas constituem uma poderosa ferramenta de produção de conteúdo para TV digital, ao unir as vantagens dos paradigmas declarativo e imperativo.

Dessa forma, para se conceber uma aplicação interativa é necessária a compreensão das diferentes possibilidades de interação deste programa. Nesse sentido, a Figura 3.1 [Silva 2010] sumariza as características dos aplicativos e programas interativos.

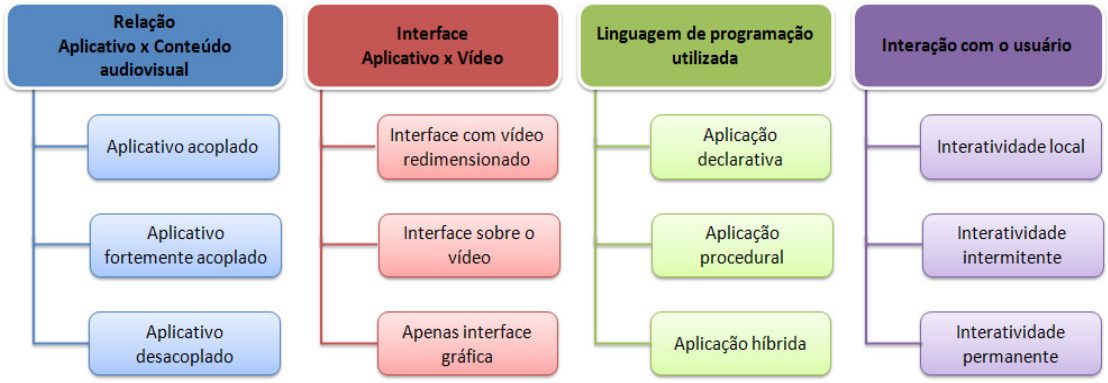


Figura 3.1 - Principais características das aplicações interativas.

Como foi apresentado nas seções anteriores, a interatividade traz uma grande novidade em relação à TV tradicional: a possibilidade vincular uma gama aplicações de computador ao conteúdo televisivo transmitido. Isto traz impactos significativos não só na forma de assistir TV, mas também na forma de produção do seu conteúdo.

Sabe-se que aplicações interativas nada mais são do que softwares; e como tal, precisam passar por um processo que garanta a execução consistente e a estruturação das atividades que compõem o trabalho da equipe de desenvolvimento, além da adequação e qualidade do produto final.

O grande desafio não está na produção das aplicações interativas em si, mas na produção dos programas de TV digital, que agregam mecanismos de interatividade ao programa de TV convencional a partir da vinculação de softwares interativos. Por esta razão, o processo de desenvolvimento de programas para TV digital integra atividades inerentes ao processo de produção de TV e atividades do desenvolvimento de software [Veiga 2007].

3.4.5. Software Tradicional x Software para TV

Em termos computacionais, uma aplicação de TV digital pode ser entendida como um software, especificamente uma aplicação multimídia, através do qual um telespectador pode interagir via controle remoto [Rodrigues 2006]. Isso significa que um telespectador pode receber do difusor, além do vídeo/áudio, softwares que possibilitam que ele interaja com o conteúdo veiculado.

De acordo com [Marques Neto 2008], os softwares para TV digital podem ser categorizados em:

1. Softwares que não têm relação com a semântica do conteúdo de áudio e vídeo apresentados, por exemplo, *e-mail* e *TV-Banking*.
2. Softwares que têm relação com a semântica do conteúdo de áudio e vídeo apresentados, mas sem restrições fortes de sincronização, por exemplo, a cotação das ações da bolsa durante um programa de economia.
3. Softwares que têm relação com a semântica do conteúdo de áudio e vídeo apresentados exibidos de forma sincronizada, por exemplo: anúncios interativos de produtos exibidos em momentos específicos de uma transmissão. Esta última categoria pode ainda ser subdividida em:

   a) Softwares com a semântica do conteúdo de áudio e vídeo conhecido a priori;

   b) Softwares com a semântica do conteúdo de áudio e vídeo gerado ao vivo;

O processo de construção de softwares da primeira categoria não se diferencia dos processos tradicionais. Assim, a única diferença de um software de TV digital para os demais está no tratamento dado aos requisitos não-funcionais tais como, plataforma de execução (receptor de TV), dispositivos de entrada (controle remoto, *smartphone*, etc).

Já a construção de softwares para TV digital pertencentes às outras categorias é feita respeitando particularidades do ambiente de TV. Essas particularidades influenciam diretamente o processo de produção e por isso, essas duas categorias devem receber dos projetistas um tratamento diferenciado. Entre as principais particularidades podem ser citadas:

1. Os softwares podem são partes do conteúdo dos programas de TV e esses, por sua vez, têm formato e contexto próprios;
2. O resultado da execução pode ser gerar elementos na tela TV, que é um espaço de uso tradicionalmente coletivo ou em dispositivos separados, como no caso do Ginga [Soares 2009].

3. Eles exigem infra-estrutura de transmissão e componentes de software/hardware adequados para o seu funcionamento.
4. Eles modificam os programas de TV tradicionais de forma a capacitá-los para lidar com diferentes níveis de interatividade e com uma organização do conteúdo não-linear.

Um processo de produção de um software específico para TV digital deve considerar tanto estas particularidades como também fornecer um suporte diferenciado a cada uma delas. O problema é que este processo não é usual nem para a indústria de TV, que não tem cultura no desenvolvimento de softwares, nem para indústria de software, que não tem cultura de desenvolvimento de conteúdo multimídia para TV.

Com a necessidade de se produzir aplicações para TV digital, os processos produtivos de programas que antes eram específicos do ambiente de TV, devem começar a agregar uma variedade de recursos oriundos de outros contextos (computação, design, etc.). A introdução de componentes interativos (softwares) nos programas de TV passa a envolver tanto profissionais de produção de TV (diretores, produtores, editores e etc.), quanto aqueles ligados à produção de software (programadores, analistas, etc.). O desafio neste caso é aliar ao processo de modelagem de componentes para o ambiente de TV, as técnicas tradicionalmente usadas na engenharia de software para solução de questões como, por exemplo, o reuso e a estruturação de requisitos.

Alguns trabalhos tratam a questão da modelagem de componentes interativos para o ambiente de TV e apontam que a rota mais curta para resolver esse desafio é adaptar algumas das abordagens usadas em modelos de processo da engenharia de software às restrições do ambiente de TV. A próxima seção, apresenta um modelo de processo de software denominado StoryToCode, que aparece como uma alternativa a este problema [Marques Neto 2009].

3.4.6. O Modelo de Processo StoryToCode

O StoryToCode é um modelo de processo de software que permite a especificação e construção de componentes para uso em programas de TV digital e em outros contextos. Para este processo, o termo contexto refere-se à plataforma de execução do software (TV, smartphone, Web, etc.). A dinâmica do modelo consiste em partir de um Storyboard/Cenário e usar conceitos de modelagem de sistemas para criar um conjunto de elementos que compõem um programa interativo. Esses elementos representam tanto as mídias, quanto os componentes de software. Essa criação é feita de forma a destacar algumas visões sistêmicas (estrutura, eventos, etc.) a fim de poder reusar os artefatos gerados em outros contextos.

A escolha do Storyboard/Cenário como ponto de partida assume as seguintes hipóteses: (1) os Storyboards/Cenários são artefatos que sempre estão presentes no processo de especificação de um programa interativo e (2) o uso dos Storyboards/Cenários facilita o entendimento e a execução de todo o processo de desenvolvimento de um programa interativo feito por uma equipe multidisciplinar. Em um ambiente real de produção de programas interativos, essa equipe poderia ser

formada por analistas de sistemas, programadores, designers, editores e diretores de TV entre outros.

A arquitetura do StoryToCode é inspirada na MDA [Perovich 2009] e está dividida em três partes relacionadas entre si, conforme a Figura 3.2:

1. Storyboard/Cenário, que é considerado na MDA como *Computational Independent Model* (CIM);
2. Diagrama de Elementos (*Components*), o qual, a depender do nível de detalhamento, pode ser chamado na MDA de *Plataform Independent Model* (PIM) ou de *Plataform Specific Model* (PSM) e;
3. Geração de Código (*Code*) para uma plataforma específica. A dinâmica dessa arquitetura define como deve ser realizada a transformação dos Storyboards/Cenários em um conjunto de elementos abstratos e, posteriormente, a transformação desses elementos em código.

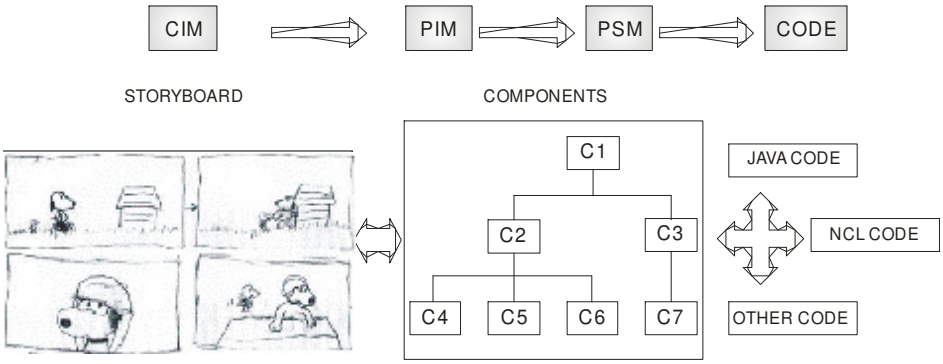


Figura 3.2 - Arquitetura do Modelo StoryToCode

Os Storyboards/Cenários presentes na primeira parte do StoryToCode são produzidos pela equipe de TV e usados como repositório para definição dos requisitos. Esses requisitos deverão ser atendidos na geração do conjunto de elementos de uma aplicação. Na versão atual do StoryToCode, essa geração não é feita de forma automatizada conforme ilustrado na Figura 3.3. Assim, cabe a equipe de projeto de software usar os Storyboards/Cenários para extrair os requisitos e gerar uma representação abstrata do conjunto de elementos contidos nas cenas de um programa, com as suas relações e os seus eventos de interatividade. Esse conjunto de elementos está representado pela segunda parte do StoryToCode.

O conjunto de elementos do StoryToCode caracterizam o *Platform Independent Model* da MDA. É importante ressaltar que esse conjunto de elementos não representa apenas um documento para auxílio no entendimento, na manutenção ou evolução do software, usualmente encontrado nos modelos conceituais para especificação de softwares. Esse conjunto é também um artefato que pode ser compilado (transformado) diretamente em outros modelos (*Platform Specific Model*) ou em códigos de linguagens de programação. Para que isso seja possível, a construção do conjunto de elementos

deve usar uma notação não ambígua e padronizada a fim de permitir a sua transformação em códigos para plataformas diferentes. O StoryToCode define uma hierarquia de elementos abstratos (Elementos do StoryToCode), representados através da notação UML e do formato XMI [Yang 2009]. Esses elementos devem ser estendidos para se adequar aos requisitos específicos de um Storyboard/Cenário.

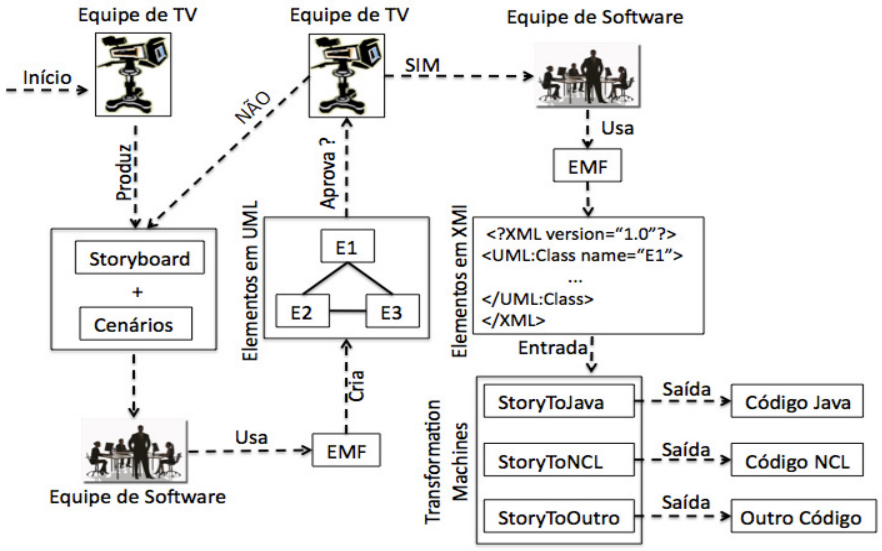


Figura 3.3 – Dinâmica do Processo de Geração de Código no StoryToCode.

O StoryToCode define que cada um dos elementos do modelo deve ser especializado para conter sua própria lista de características a fim de poder abstrair corretamente um Storyboard/Cenário. A transformação inicial dos elementos (especialização) antes da geração de código caracteriza o conceito de *Platform Specific Model* da MDA. Essa capacidade de extensão do modelo permite adequá-lo a qualquer especificação tanto no que diz respeito à estruturação das mídias, quanto dos componentes de software dos Storyboards/Cenários de um programa interativo. A possibilidade de descrever cada um dos elementos que compõem um Storyboard/Cenário com riqueza de detalhes é que permite o uso das transformações para ir reduzindo o nível de abstração até que o código de uma aplicação para um contexto específico seja obtido. Para permitir a especificação/ extensão dos elementos em UML e também em XMI, o StoryToCode usa o arcabouço denominado de *Eclipse Modeling Framework* (EMF) [Steinberg 2009]. Esse arcabouço contém um *parser* XML que permite gerar código XMI a partir de especificações visuais feitas em UML.

Uma vez que o conjunto de elementos esteja definido pela equipe de software e aprovado pela equipe de TV, chega-se a terceira etapa do StoryToCode: geração de código. A Figura 3.3 exibe toda a dinâmica do processo de geração de código a partir de Storyboards/Cenários no modelo StoryToCode. O objetivo específico da terceira etapa é gerar o código de uma aplicação para um contexto específico a partir do conjunto de elementos genéricos gerado na etapa anterior. Para isso, o StoryToCode define um componente especial chamado de *transformation machine* (um *parser*). Ele recebe

como entrada o conjunto de elementos (em XMI) e gera como saída o código para uma plataforma específica.

O primeiro passo dessa etapa do StoryToCode, é usar o EMF na tarefa de obter automaticamente os dados de entrada (código XMI dos elementos representados em UML) para um *transformation machine*. O uso do formato XMI, que é independente de plataforma, é um requisito de entrada para o processo de geração de código no StoryToCode. Assim, depois de gerar o arquivo XMI e de carregá-lo na memória, a equipe de software escolhe uma das instâncias disponíveis de *transformation machine* para gerar automaticamente o código fonte em uma plataforma específica. Cada instância contem um conjunto de regras de transformação, baseadas no conhecimento de origem (conjunto de elementos), e da estrutura dos elementos de destino (código para uma plataforma específica). Um *transformation machine* permite adicionar regras (e outras informações importantes) para permitir mapear um elemento no seu código fonte correspondente em uma linguagem de programação. Assim, um único conjunto de elementos pode ser transformado em código para plataformas diferentes. Para isso deve existir uma instância do *transformation machine* específica para cada plataforma.

As principais vantagens do uso do StoryToCode são:

- Permitir o projeto e implementação de uma aplicação (conjunto de componentes interativos), independente do contexto, levando em consideração as particularidades de um programa interativo de TV;
- Diminuir a responsabilidade do gerador de conteúdo através da descentralização das etapas de produção que estão fora do seu universo de trabalho original: a especificação e implementação de um artefato de software;
- Permitir a participação de outros atores (analistas de sistemas, programadores, designers e etc.) no processo produtivo de um programa interativo de TV;
- Diminuir do esforço despendido durante o processo de produção dos componentes interativos.
- Permitir o reuso dos componentes de software, criados para TV, em outros ambientes.
- Permitir o reuso de componentes de software, criados em outros domínios, no ambiente de TV.

Após a análise detalhada das questões ligadas à concepção de aplicações interativas para TV digital apresentada nas primeiras seções, a sequência do texto irá tratar da infraestrutura disponível no receptor de TV para a execução dessas aplicações.

3.5. O *middleware* Ginga

O Ginga é a especificação de *middleware* do SBTVD, resultado da integração das propostas FlexTV [Leite 2005] e MAESTRO [Soares 2006], desenvolvidas por consórcios liderados pela UFPB e PUC-Rio no projeto SBTVD , respectivamente.

O FlexTV, proposta inicial de *middleware* procedural do SBTVD, incluía um conjunto de APIs compatíveis com outros padrões além de funcionalidades inovadoras, como a possibilidade de comunicação com múltiplos dispositivos, permitindo que

diferentes usuários pudessem interagir com uma mesma aplicação interativa a partir de dispositivos remotos. Já o MAESTRO foi a proposta inicial de *middleware* declarativo do SBTVD. O foco era oferecer facilidade do sincronismo espaço-temporal entre objetos multimídia, utiliza a linguagem declarativa NCL e agregar as funcionalidades da linguagem de script da linguagem Lua.

O Ginga integrou estas duas soluções, chamadas de Ginga-J [Souza Filho 2007] e Ginga-NCL [Soares 2007], tomando por base as recomendações internacionais da ITU [ITU J200 2001]. Desta forma, o Ginga é subdividido em dois subsistemas interligados, também chamados de Máquina de Execução (Ginga-J) e Máquina de Apresentação (Ginga-NCL) conforme ilustrado na Figura 3.4. O conteúdo imperativo é executado por uma Máquina Virtual Java (JVM).

Figura 3.4 – Visão geral do *middleware* Ginga

Outro aspecto importante é que os dois subsistemas do Ginga não são necessariamente independentes, uma vez que a recomendação do ITU inclui uma "ponte", que deve disponibilizar mecanismos para intercomunicação entre os mesmos, de um modo que as aplicações imperativas utilizem serviços disponíveis nas aplicações declarativas, e vice-versa. Portanto, é possível a execução de aplicações híbridas em um nível acima da camada dos ambientes de execução e apresentação, permitindo agregar as facilidades de apresentação e sincronização de elementos multimídias da linguagem NCL com o poder da linguagem orientada a objetos Java.

O Núcleo Comum Ginga (*Ginga Common Core*) é o subsistema do Ginga responsável por oferecer funcionalidades específicas de TV Digital comuns para os ambientes imperativo e declarativo, abstraindo as características específicas de plataforma e hardware para as outras camadas acima. Como suas principais funções, podemos citar: a exibição e controle de mídias; o controle de recursos do sistema (o canal de retorno, dispositivos de armazenamento); acesso a informações de serviço; sintonização de canais, entre outros.

3.6. Ginga-J: O Ambiente Imperativo do Ginga

3.6.1. Java e TV Digital

Java foi criada pela *Sun Microsystems* e define um conjunto de tecnologias, que inclui desde a linguagem de programação, um conjunto de classes organizadas em bibliotecas (a API de Java), um formato de arquivo para as classes (arquivo *class*) e uma máquina virtual, até os ambientes de execução e desenvolvimento, também chamada de plataforma Java

Java pode ser utilizada para se criar vários tipos de aplicativos, desde aplicações *Desktop* até aplicações designadas para serem controladas pelo software que as executa, tais como *Applets* que são carregados pela Web e executados dentro de um browser, *Servlets*, que são designados para serem executados dentro de um servidor de aplicações Web; *Midlets*, designados para serem executados dentro de dispositivos móveis. Para o ambiente de TVD existem os *Xlets*, que são aplicações que são executadas por um *middleware* em terminais de acesso que suportam a API JavaTV.

Uma das propriedades fundamentais que torna Java apropriada como linguagem de programação para TV Digital é a habilidade de construir códigos móveis que podem rodar em máquinas pequenas. A Figura 3.5 ilustra este cenário. O código fonte de uma aplicação para TV Digital geralmente é produzido na emissora usando um ambiente de desenvolvimento integrado (do inglês, IDE *Integrated Development Environment*), o qual muitas vezes torna possível gerar códigos de forma rápida. Tipicamente, a IDE possibilita aos desenvolvedores compilar o código em arquivos de classes móveis. Esses arquivos são transportados para os terminais de acesso utilizando os protocolos de difusão, os quais usam uma máquina virtual Java, especificamente um *ClassLoader*, no *middleware* para executá-los. Uma vez instanciado, o código fonte das aplicações é capaz de acessar classes Java de APIs da JVM ou então acessar código nativo através de *Java Native Interface* (JNI), que dá acesso aos recursos de hardware da plataforma.

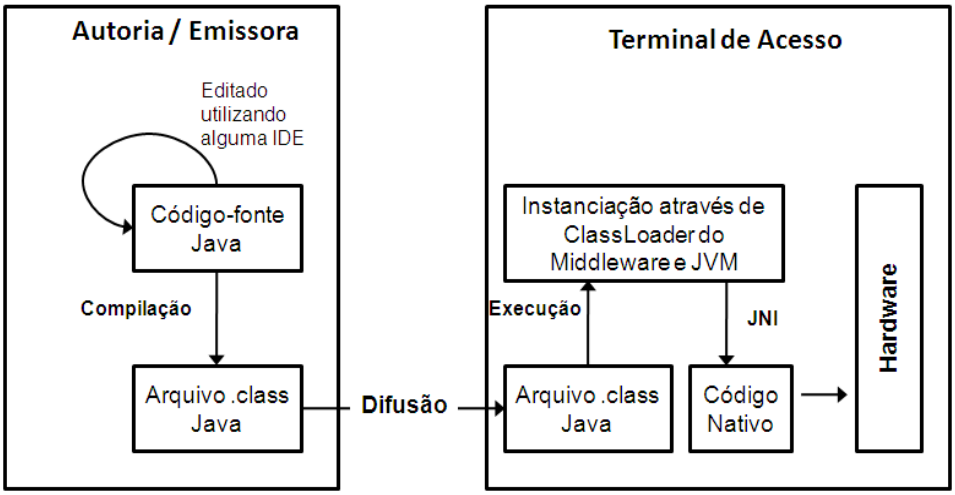


Figura 3.5 – Modelo de programação Java para TV Digital

3.6.2. Visão geral do Ginga-J

Como todo ambiente Java, o Ginga-J é composto por um conjunto de APIs, conforme mostra a Figura 3.6. Tais APIs são definidas para atender funcionalidades necessárias para o desenvolvimento de aplicativos para TVD, desde a manipulação de dados multimídia até protocolos de acesso. Sua especificação é formada por uma adaptação da API de acesso a informação de serviço do padrão japonês [ARIB STD-B.23 2006], pela especificação Java DTV [JavaDTV API 2009], que inclui a API JavaTV, além de um conjunto de APIs adicionais de extensão ou inovação.

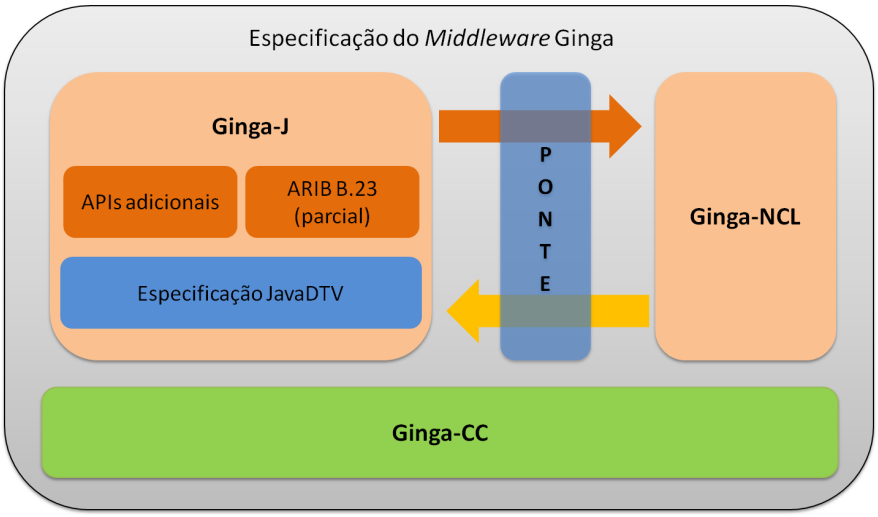


Figura 3.6 – Visão geral do Ginga-J

As APIs adicionais incluem um conjunto de classes disponíveis para a ponte entre os aplicativos NCL e Java, funcionalidades adicionais para sintonia de canais, envio de mensagens assíncronas pelo canal de interatividade e integração de dispositivos externos ao *middleware*, viabilizando a interação simultânea de múltiplos usuários e dispositivos em aplicações de TVD [Silva 2007, Silva 2008].

A especificação Java DTV é uma plataforma aberta e interoperável que permite a implementação de serviços interativos com a linguagem Java, tendo sido inserida recentemente ao conjunto de APIs do Ginga-J. Funcionalmente, a JavaDTV substitui a coleção de APIs definidas inicialmente para o Ginga-J [Souza Filho 2007] e utilizadas pelo padrão GEM (*Globally Executable MHP*) [ETSI TS 102 819], tais como DVB (*Digital Video Broadcast*), DAVIC (*Digital Audio Video Council*) e HAVi (*Home Audio Video Interoperability*). O objetivo da JavaDTV é fornecer uma solução livre de *royalties* para permitir a fabricação de aparelhos de TVD e/ou conversores e desenvolvimento de aplicações com um custo mais acessível. Uma diferença importante da Java DTV em relação ao GEM, é a API LWUIT *(LightWeight User Interface Toolkit*), responsável por definir elementos gráficos, extensões gráficas para TV digital, gerenciadores de *layout* e eventos do usuário.

Adicionalmente, o Ginga-J é composto pela API JavaTV e pelo ambiente de execução Java para sistemas embarcados (JavaME), incluindo a plataforma CDC

(*Connected Device Configuration*), e as APIs dos perfis: FP (*Foundation Profile*) e PBP (*Personal Basis Profile*), conforme mostrado na Figura 3.7.

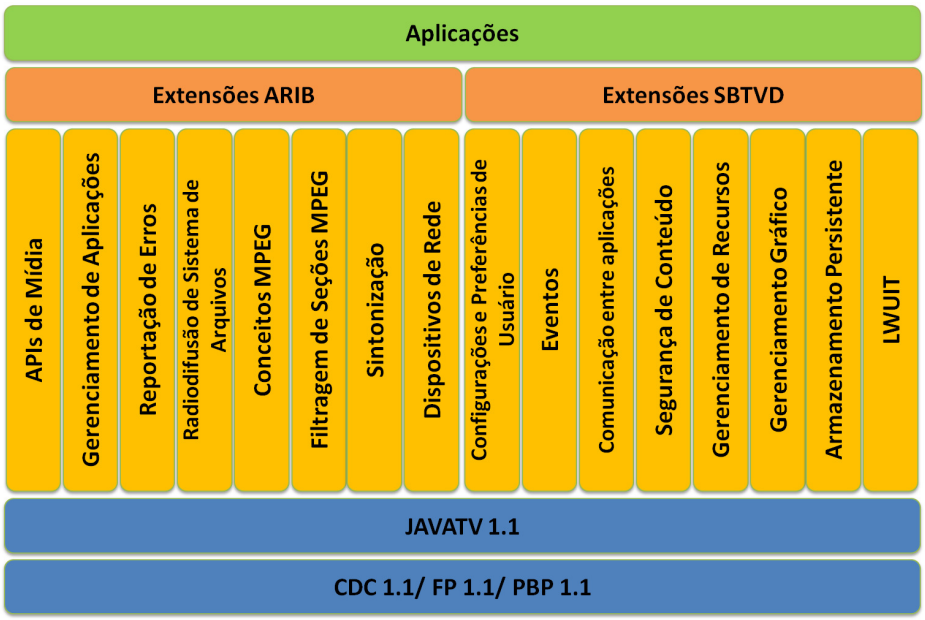


Figura 3.7 – Conjunto de APIs Ginga-J

3.6.3. Pacotes do Ginga-J

A norma Ginga-J [NBR 15606-4 2010] define uma plataforma para execução de aplicações Java para TV Digital. Assim como outras tecnologias Java, uma forma de estudar este ambiente é através das suas APIs, representadas por conjuntos de pacotes. Tais conjuntos são divididos em 7 (sete) partes:

1. Pacotes da plataforma básica Java – representa as funcionalidades de um ambiente Java básico um sistema embarcado baseado nos ambientes CDC [CDC 1.1 2008], FP [FP 1.1 2008] e PBP [PBP 1.1 2008].
2. Pacotes da especificação JSSE 1.0.1 – implementa funcionalidades opcionais de segurança para a plataforma básica de Java para TV Digital, como por exemplo protocolos de transporte seguro [JSSE 1.0.1 2006].
3. Pacotes da especificação JCE 1.0 – implementa outras funcionalidades opcionais de segurança para a plataforma básica de Java para TV Digital, especificamente para operações de criptografia [JCE 1.0.1 2006].
4. Pacotes da especificação SATSA 1.0.1 – permite comunicação com dispositivos externos (normalmente *smartcards*) utilizando o protocolo APDU (do inglês, *Application Protocol Data Unit*) [SATSA 1.0.1 2007].
5. Pacotes da especificação JavaTV 1.1 – implementa o modelo de gerenciamento de aplicações, funcionalidades específicas para TV Digital num grau de abstração maior, além de incluir a API JMF *(Java Media Framewor)* [JavaTV 1.1 2008].

6. Pacotes da especificação JavaDTV 1.3 – extende os pacotes do JavaTV 1.1 para implementar funcionalidades específicas de TV Digital adicionais ou de menor grau de abstração. Também contém os pacotes de APIs gráficas do LWUIT (componentes gráficos, tratamento de eventos do usuário e gerenciador de layout) [JavaDTV 1.3 2009].
7. Pacotes específicos Ginga-J – contém pacotes que implementam funcionalidades exclusivas do sistema brasileiro (controle de planos gráficos, ou que foram herdadas do sistema japonês (acesso a informações de serviço dependente de protocolo) [NBR 15606-4 2010].

As informações mais técnicas sobre cada pacote dessas especificações podem ser encontradas nas referências. A norma Ginga-J [NBR 15606-4 2010] também possui várias descrições desses pacotes.

3.6.4. Emulador Ginga-J

O Emulador Ginga-J[6] é um projeto de código aberto lançado pelo LAVID (Laboratório de Aplicações de Vídeo Digital) da UFPB. Ele consiste num ambiente para execução de aplicações Java para TVD que já é compatível com a nova especificação de APIs do Ginga-J. Por exemplo, esta ferramenta já implementa elementos gráficos e gerenciadores de layout da API LWUIT e vários pacotes da API JavaDTV.

Desenvolvido exclusivamente na linguagem Java, o objetivo do emulador é oferecer um ambiente de execução de aplicações mais simples de instalar e usar num computador pessoal. Porém, devido essa característica a ferramenta apresenta algumas limitações para representar todas as funcionalidades disponíveis num terminal de acesso real. A principal está relacionada a exibição de vídeo e áudio com formatos da alta resolução ou taxa de dados, uma vez que o processamento das mídias é realizada exclusivamente através de software. Outra restrição está relacionada ao uso dos protocolos das redes de difusão, como sinalização de aplicações, seleção de fluxos elementares, acesso a informações de serviço e carrossel de dados, já que a execução das aplicações é realizada localmente. Entretanto, através do emulador é possível realizar prototipação rápida da interface gráfica do usuário para aplicações de TVD, permitindo testar aspectos de disposição, cores, forma, navegação e interação com o usuário.

A arquitetura do Emulador Ginga-J é baseada no Xletview[7], que implementa APIs do GEM. Foram reutilizados os seguintes elementos: (1) interface gráfica com o usuário; (2) gerenciador de aplicações; (3) tratamento de eventos do usuário; (4) modelo das camadas gráficas de um terminal de acesso.

O elemento (1) da arquitetura, ilustrado na Figura 3.8, consiste de um ambiente com: (a) uma área para execução das aplicações que emula uma visor de TV; (b) um controle remoto virtual; (c) menu para acesso as funcionalidades do gerenciador de aplicações.

---

[6] Emulador Ginga-J– Disponível em: http://dev.openginga.org/projects/gingaj-emulator

[7] XletView – Disponível em: http://www.xletview.org

Para o elemento (2) foi possível reaproveitar todo o código, pois o Ginga-J utiliza o mesmo modelo de aplicações (JavaTV Xlets) do GEM.

Para os elementos (3) e (4) da arquitetura, o reuso de código foi possível pois o XletView utiliza as bibliotecas AWT e Swing do ambiente gráfico do *Java Standard Edition* (JSE). No caso, é utilizado mecanismo de tratamento de eventos de eventos do AWT para captura e notificações de eventos do usuário (através de cliques no controle remoto virtual ou pressionamento de teclas do teclado do computador) e a classe ***java.swing.JLayedPlane*** para representar as camadas de planos gráficos de um terminal de acesso.

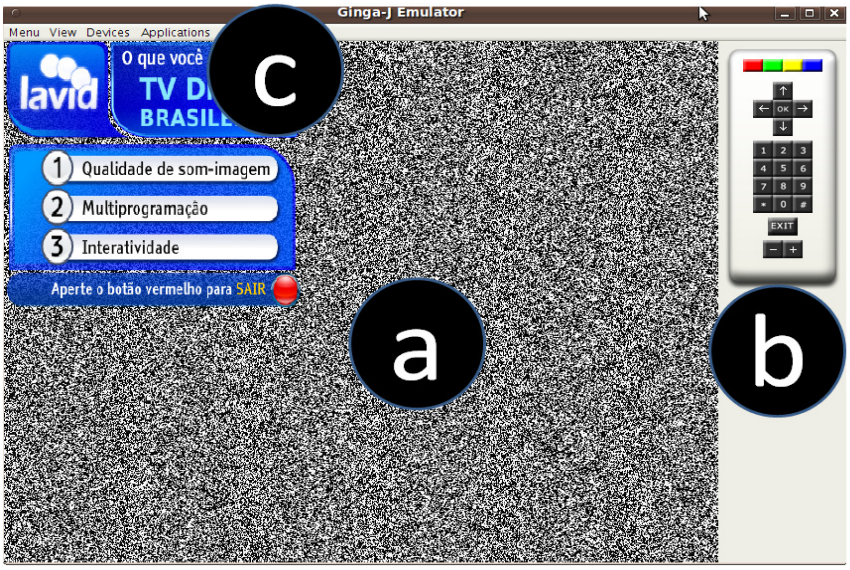


Figura 3.8 – Interface gráfica do Emulador Ginga-J

Outra estratégia de desenvolvimento empregada foi a substituição de artefatos (classes e interfaces) do *XletView* aderentes ao GEM por elementos de equivalência funcional aderentes ao JavaDTV no Emulador Ginga-J. Em outras palavras, algumas classes/interfaces do Ginga-J apenas mudam de nome de classe e assinaturas de métodos em relação ao GEM, tornado possível a reutilização de métodos privados e trechos de código de método. Nesse processo a maior dificuldade foi retirar dependências (através de herança, composição, agregação, referência direta etc.) entre classes do GEM e pacotes internos do XletView. Contudo, para alguns pacotes do JavaDTV foi necessária um implementação sem nenhum reuso do código, como por exemplo as APIs do LWUIT, que não possui equivalência funcional com HAVi (utilizado no *XletView*).

Como requisitos adicionais implementados no Emulador Ginga-J podemos citar: (1) melhor integração da camada gráfica de vídeo com a camada gráfica da aplicação (diminuição do efeito de *flicker*); (2) capacidade de reprodução de múltiplos arquivos de áudio e vídeos nas aplicações; (3) possibilidade de configurar e trocar os vídeos da camada gráfica de vídeo que representam um canal de TV e (4) integração com dispositivos externos. Este último requisito permite o desenvolvimento de aplicações

utilizando a API de Integração de Dispositivos[8][Silva 2007] e é relevante, uma vez que não está presente em nenhum outro ambiente de execução de aplicações para TVD disponível atualmente.

3.6.5. Exemplo de Aplicação Ginga-J

Esta seção procura explorar um exemplo da construção de uma aplicação imperativa utilizando o Ginga-J. A aplicação consiste num serviço que permite obter informações adicionais do conteúdo exibido no programa (similar aos menus interativos disponíveis em DVDs). A figuras 3.9 e 3.10 ilustram a tela inicial (lado esquerdo) e a tela com o menu de opções da aplicação (lado direito), respectivamente.

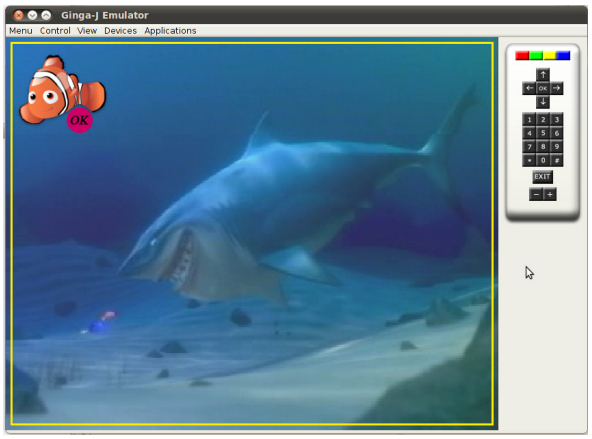


Figura 3.9 – "Tela Inicial"

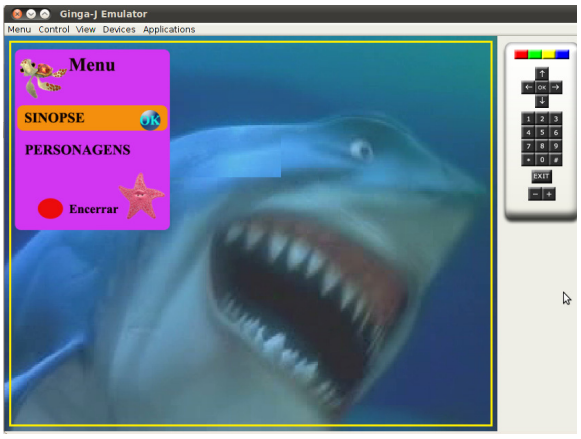


Figura 3.10 – "Tela Menu 1"

A seguir são descritos os passos definidos para o desenvolvimento da aplicação. O objetivo é explicar de forma geral o desenvolvimento de uma aplicação Ginga-J, o código-fonte e a documentação técnica completa do exemplo estão disponíveis na página do projeto do Emulador Ginga-J (ver nota de rodapé 6 na página 18).

Passo 1- Definição de um Xlet

Todas as aplicações Ginga-J devem conter uma classe implementando a interface ***javax.microedition.xlet.Xlet*** (ver PBP 1.1 2008), que deve ser instanciada de acordo com as definições de sinalização de aplicação (ver NBR 15606-3:2007, 12.16). Caso contrário, a classe (o objeto que representa a instância da aplicação) será ignorada quando enviado para um terminal de acesso. Dessa forma, é possível executar a aplicação num ambiente orientado a serviços e mantidas por um gerenciador de aplicações do *middleware*, que garante cada aplicação acesse seu ambiente de execução através de um contexto de serviço (representado por uma instância da classe ***javax.microedition.xlet.ServiceContext***).

Considerando que a aplicação define uma classe que implementa ***javax.microedition.xlet.Xlet***, esta classe deve conter no mínimo 4 (quatro) métodos da interface que permite que a plataforma gerencie seu ciclo de vida e envie informações

[8] É importante lembrar que a API de *Interaction Devices* ainda está em discussão como proposta para o Ginga-J no Fórum SBTVD e ainda não aprovada até o momento de escrita desse texto.

ou eventos de mudanças. O trecho de código a seguir a estrutura inicial de qualquer aplicação Ginga-J: uma classe que implementa 1 (uma) interface e contém obrigatoriamente 4 (quatro) métodos.

```
import javax.microedition.xlet.*;

public class Xlet1 implements Xlet{
   private XletContext context = null;

   public void initXlet(XletContext xletContext)
        throws XletStateChangeException {}

   public void startXlet() throws XletStateChangeException {}

   public void pauseXlet(){}

   public void destroyXlet(boolean flag) throws XletStateChangeException {       }
}
```

A partir daí é necessário implementar cada método (***initXlet, startXlet, pauseXlet e destroyXlet***) de acordo com o seu papel no ciclo de vida da aplicação. Detalhes sobre o papel de cada método podem ser encontrados na norma Ginga-J ou na API JavaTV.

Passo 2- Configuração dos planos gráficos

As aplicações Ginga-J conseguem obter acesso de forma genérica aos planos gráficos oferecidos pelo terminal de acesso, para configuração e exibição de conteúdo de acordo com um modelo de camadas na tela do dispositivo. De acordo com a NBR 15606-1 (2010), a organização dos planos ou camadas gráficas é feita conforme a Figura 3.11.

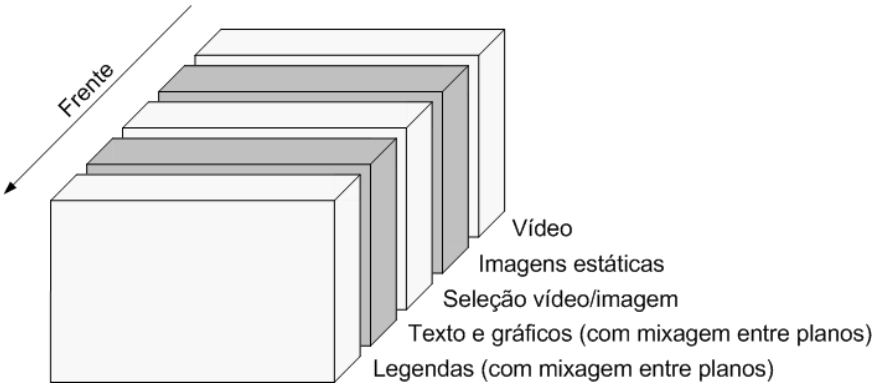


Figura 3.11 – Estrutura de camadas gráficas de um terminal de acesso.
Fonte: [NBR 15606-1 2010]

Com exceção do plano de legendas, todos os planos são acessíveis por aplicações Ginga-J, sendo uma característica nativa de um terminal de acesso. Portanto, existem 4 (quatro) planos sobre os quais uma aplicação Java pode operar:

1. Plano[0]: Plano de texto e gráficos;
2. Plano[1]: Plano de seleção vídeo/imagem;

3. Plano[2]: Plano de imagens estáticas;
4. Plano[3]: Plano de vídeo.

Para cada um destes planos, é permitido obter suas características e efetuar operações gráficas sobre eles. O Ginga-J, através de pacotes da JavaDTV e pacotes específicos SBTVD, define abstrações (classes e interfaces) que implementam este modelo gráfico do terminal de acesso. A Figura 3.12 resume esses elementos:

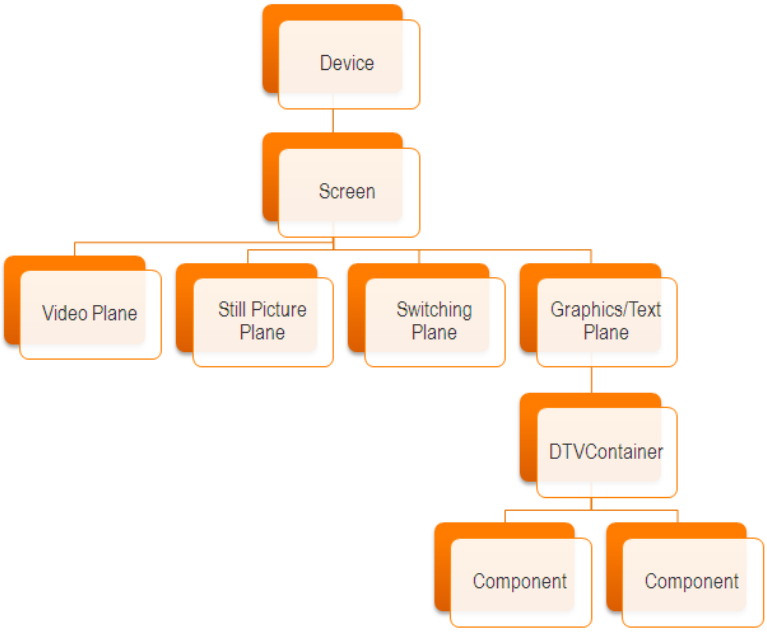


Figura 3.12 – Hierarquia de abstrações representando o modelo gráfico no Ginga-J

A classe ***com.sun.dtv.ui.Device*** (representa o dispositivo no qual o Ginga está instalado) dá acesso a uma ou mais classes ***com.sun.dtv.ui.Screen*** (representa cada tela disponível para exibição dos gráficos da aplicação). Esta classe permite acessar todos os planos do terminal, retornando instancias da classe ***com.sun.dtv.ui.Plane*** com os respectivos identificadores específicos (***Video Plane, Still Picture Plane, Switching Plane e Graphics/Text Plane***). Para cada plano é possível verificar suas características (***com.sun.dtv.ui.Capabilities***) e obter uma classe ***com.sun.dtv.ui.DTVContainer*** que é o componente que suporta todos os tipos de ***com.sun.dtv.lwuit.component*** e operações gráficas definidas na API do LWUIT (por exemplo, ***com.sun.dtv.lwuit.Form***).

A classe ***ResourceComponents*** é apenas uma classe auxiliar (pertence a própria aplicação e não faz parte da especificação Ginga-J) para encapsular as classes que permitem configurar (***Device, Screen, Plane*** e ***Capabilities***) e desenhar elementos gráficos na tela do terminal do acesso (***Form*** e ***DTVContainer***), conforme ilustrado no trecho de código a seguir.

```
public void initXlet() throws XletStateChangeException {

               ResourceComponents.init();
               Device device = Device.getInstance();
               Screen currentScreen = device.getDefaultScreen();

               ResourceComponents.manager.addUserInputEventListener(this,
                    ResourceComponents.anyColoredKeyTyped);

               Plane[] planes = currentScreen.getAllPlanes();

               for(int i=0; i<planes.length; i++) {
                      Capabilities cap = planes[i].getCapabilities();
                         if (cap.isGraphicsRenderingSupported()) {
                                    ResourceComponents.plane = planes[i];
                                    ResourceComponents.planeSetup =
                                       ResourceComponents.plane.getCurrentSetup();
                                    break;
                         }
               }
               ResourceComponents.form = new Form();
               ResourceComponents.dtvcontainer =
               DTVContainer.getDTVContainer(ResourceComponents.plane);
               ResourceComponents.dtvcontainer.setLayout(null);
               ResourceComponents.dtvcontainer.addComponent(ResourceComponents.form);
               ResourceComponents.dtvcontainer.setVisible(true); (…)
```

Passo 3 – Reserva e configuração de recursos limitados

A API JavaDTV fornece, através do pacote ***com.sun.dtv.resources***, um *framework* básico para gerenciamento de recursos limitados em terminais de acesso. Consequentemente, para acessar qualquer recurso limitado é necessário fazer uma solicitação de reserva previamente e após o seu uso ou configuração realizar uma liberação. É importante lembrar que a aplicação também deve possuir a permissão para acessar determinados recursos.

As seguintes classes são recursos limitados, ou seja implementam a interface ***com.sun.dtv.resources ScarceResource*** no Ginga-J: filtros de seções MPEG-2 (***com.sun.dtv.filtering.DataSectionFilterCollection***), eventos de entrada do usuário (***com.sun.dtv.ui.event.UserInputEvent, com.sun.dtv.ui.event.KeyEvent, com.sun.dtv.ui.event.MouseEvent, com.sun.dtv.ui.event.RemoteControlEven***), dispositivos de rede (***com.sun.dtv.net.NetworkDevice***), telas (***com.sun.dtv.ui.Screen***) e sintonizador (***com.sun.dtv.tuner.Tuner***).

O trecho de código abaixo exemplifica a reserva da escuta de eventos de qualquer tecla colorida do controle remoto (através da chamada do método ***reserve***()). No exemplo, ***anyColoredKeyTyped*** é uma instância da classe ***com.sun.dtv.ui.event.RemoteControlEvent***.

```
ResourceComponents.anyColoredKeyTyped =
               new RemoteControlEvent(null, java.awt.event.KeyEvent.KEY_TYPED,0, 0,
                    RemoteControlEvent.VK_COLORED, KeyEvent.CHAR_UNDEFINED);
try {
       ResourceComponents.anyColoredKeyTyped.reserve(true, -1, null);
} catch (…)
```

Passo 4 – Tratamento de Eventos de Entrada do Usuário

Depois de configurar as camadas gráficas para desenho dos elementos da interface gráfica do usuário e realizar a reserva de recursos limitados do terminal, o próximo passo é definir o esquema de navegação de aplicação através de tratamento de eventos de entrada do usuário.

No Ginga-J o mecanismo de tratamento de eventos de entrada do usuário é oferecido através de componentes específicos de TVD (pacote ***com.sun.dtv.ui.event***) e componentes mais gerais providos pela LWUIT (pacote ***com.sun.dtv.lwuit.events***). A classe que trata os eventos gerados por componente gráficos é a ***com.sun.dtv.ui.event.UserInputEventManager***.

Uma boa prática é definir uma classe para cada "tela" da aplicação e nela encapsular o tratamento dos eventos do usuário que vai definir o comportamento da aplicação de acordo com o evento de entrada do usuário (por exemplo, ir para a próxima "tela" ou voltar para a "tela" anterior).

Na sequência do texto trechos de código da tela inicial da aplicação exemplo são ilustrados. As 5 (cinco) primeiras linhas do trecho de código servem para a criação de um objeto ***UserInputEventManager*** a partir de uma referência para uma ***Screen*** (***currentScreen***) e o cadastro (método ***addUserInputEventListener)*** do objeto corrente (o objeto da Tela Inicial) como ***Listener*** (implementa a interface ***com.sun.dtv.ui.event.UserInputEventListener***) do ***UserInputEventManager*** e configurado para receber apenas eventos das teclas coloridas (já explicado no passo 3 deste seção). Já o método ***userInputEventReceived*** é definido na interface do ***Listener*** e é chamado quando algum evento é disparado em componentes gráficos da aplicação. No exemplo é verificado se o evento gerado é de uma tecla de ENTER. Caso afirmativo, a "Tela" atual é apagada e se abre uma nova "Tela" (representado pela classe ***Menu1***).

```
(…)
ResourceComponents.manager =
       UserInputEventManager.getUserInputEventManager(currentScreen);
ResourceComponents.manager.addUserInputEventListener((UserInputEventListener) this,
ResourceComponents.anyColoredKeyTyped);

(…)

public void userInputEventReceived(UserInputEvent inputEvent) {
        com.sun.dtv.ui.event.KeyEvent e = (com.sun.dtv.ui.event.KeyEvent) inputEvent;
        int type=e.getID();
        int code = e.getKeyCode();

        if (type == KeyEvent.KEY_PRESSED) {
              if (code == KeyEvent.VK_ENTER) {
                       this.clear();
                        Menu1 menu1 = new Menu1();
                        menu1.init();
               }
        }
}
```

Passo 5 – Desenho da interface gráfica com o usuário

A etapa final da construção da aplicação consiste em desenhar os elementos gráficos da aplicação nas "Telas" de navegação.

O Ginga-J, através do pacote ***com.sun.dtv.lwuit***, define o mecanismo de composição de elementos gráficos (***com.sun.dtv.lwuit.Componen***t e ***com.sun.dtv.lwuit.Container***) da API LWUIT, que segue um modelo idêntico ao das APIs AWT/Swing. Entretanto, ao contrário do AWT/Swing, não é empregado um sistema de janelas de tela cheia. Neste caso é utilizado o modelo de planos gráficos apresentado no passo 2 desta seção. As seguintes interfaces e classes devem ser empregadas seguindo a especificação JavaDTV:

Tabela 2 – Elementos do pacote com.sun.dtv.lwuit
Fonte: [NBR 15606-4 2010]

Classes	Descrição
Classe ***MediaComponent***	Possibilita a inserção e controle de conteúdo de mídia rica
Classe ***StaticAnimation***	Uma imagem capaz de animação
Classe ***Graphics***	Abstrai a plataforma de contexto gráfico, permitindo a portabilidade entre dispositivos.
Classe ***Container***	Implementa o padrão de projeto ***Composite*** para ***com.sun.dtv.lwuit.Component*** de um modo que permite arranjar e relacionar ***components*** utilizando um arquitetura de gerenciadores de layout plugáveis
Classe ***ComboBox***	Elemento gráfico que representa uma lista que permite apenas uma seleção por vez através da escolha do usuário
Classe ***Font***	Uma simples abstração de fontes de plataforma e de biblioteca que permite o uso de fontes que não são suportadas pelo dispositivo
Interface ***Painter***	Esta interface pode ser usada para desenhar em componentes de fundo de tela
Classe ***RadioButton***	Tipo específico de ***com.sun.dtv.lwuit.Button*** que mantém um estado de seleção exclusivamente de um ***com.sun.dtv.lwuit.ButtonGroup***.
Classe ***Calendar***	Possibilita a seleção de valores de data e hora
Classe ***TabbedPane***	Permite ao usuário alternar entre um grupo de componentes clicando em um guia com um determinado título e/ou ícone
Classe ***Command***	Ação referente aos "*soft buttons*" e menu do dispositivo, similar à abstração de ações do Swing
Classe ***CheckBox***	Botão que pode ser marcado ou desmarcado e ao mesmo tempo exibir seu estado para o usuário
Classe ***Form***	Componente de alto nível que é classe base para interfaces gráficas com o usuário da LWUIT. O contêiner é dividido em três partes: ***Title*** (barra de títulos localizada normalmente na parte superior), ***ContentPane*** (espaço central para disposição de elementos entre ***Title*** e ***MenuBar***) e ***MenuBar*** (barra de menu localizada normalmente na parte inferior)
Classe ***Dialog***	Um tipo de ***Form*** que ocupa uma parte da tela e aparece como uma entidade modal para o desenvolvedor
Classe ***Image***	Abstração da plataforma que trata imagens, permitindo manipulá-

	las como objetos uniformes
Classe ***TextField***	Componente para recebimento de entrada de texto de usuário que utiliza uma API mais leve, sem utilizar o suporte nativo para texto do dispositivo
Classe ***ButtonGroup***	Esta classe é utilizada para criar um escopo de múltipla exclusão para um conjunto de ***RadioButtons***
Classe ***Label***	Permite exibir *labels* e imagens com diferentes opções de alinhamento, também funciona como classe base para alinhamento da disposição de componentes
Classe ***Button***	Componente base para outros elementos gráficos que são clicáveis
Classe ***List***	Um conjunto (lista) de elementos que são criados utilizando uma ***ListCellRenderer*** e são extraídos através da ***ListModel***
Classe ***Component***	Classe base para todos os elementos gráficos da LWUIT. Utiliza o padrão de projeto ***Composite*** de forma semelhante à relação de ***Container*** e ***Component*** do AWT
Classe ***TextArea***	Componente gráfico que permite entrada de textos com múltiplas linhas editáveis, também permite exibir e editar o texto
Classe ***AWTComponent***	Estende a classe ***com.sun.dtv.lwuit.Component*** como uma variante especial que delega as ações de renderização para ***java.awt.Component***

A classe que implementa a "Tela" Menu1 da aplicação exemplo é mostrada no trecho de código a seguir. Ela é construída e configurada uma classe ***com.sun.dtv.lwuit.Label*** que possui uma ícone representado por uma instancia da classe ***com.sun.dtv.lwuit.Image***. Por fim, o ***labelMenu1*** é adicionado a uma instância de ***com.sun.dtv.lwuit.Form*** que por sua vez é exibido na tela (***método repaint()***).

```
import com.sun.dtv.lwuit.Form;
import com.sun.dtv.lwuit.Image;
import com.sun.dtv.lwuit.Label;
(…)
        Label labelMenu1;

        FileInputStream menu1 = new FileInputStream( new File(Imagens.menu1) );

                                labelMenu1 = new Label();
                                labelMenu1.setIcon( Image.createImage(menu1) );

                labelMenu1.setX(0);
                labelMenu1.setY(0);
                labelMenu1.setSize(new Dimension(Imagens.WIDTH ,Imagens.HEIGTH));

                form.removeAll();
                form.addComponent(ResourceComponents.labelMenu1);
                form.repaint();
(…)
```

3.7. Comentários Finais

Este capítulo descreveu uma introdução ao desenvolvimento de aplicações imperativas para TV Digital utilizando o Ginga. Foram apresentados conceitos de TV Interativa, convergência digital, os principais componentes de sistema de TV Digital, tipos aplicações, processo e metodologias de desenvolvimento de software na área, o *middleware* Ginga, o Emulador Ginga-J e, na última seção, a construção passo a passo

de uma aplicação Java para TV Digital. O objetivo não foi esgotar o assunto, mas levantar questões importantes e divulgar a tecnologia.

Como foi observado no decorrer no capítulo, o desenvolvimento de software para TV Digital traz consigo vários desafios e oportunidades para os desenvolvedores de software. O mercado caracterizado por produtores de conteúdo audiovisual e produtores de software devera endereçar esses novos desafios.

Em contraste a maioria dos softwares para computador, as aplicações interativas precisam de um conjunto de componentes complexos que precisam trabalhar sob uma série de circunstâncias específicas, de modo que estes softwares precisam de novos modelos de processo de desenvolvimento e novas ferramentas para endereçar suas novas necessidades.

Nesse sentido, se torna prioritário um esforço de desenvolvimento de processos e ferramentas da Engenharia de Software aplicados ao contexto de aplicações interativas. [Gawlinski 2007] caracteriza essa problemática na seguinte afirmação:

> *O problema com as aplicações de televisão interativa é que elas têm toda a complexidade das aplicações de computadores, mas estão rodando em uma plataforma que normalmente não tem problemas técnicos ou de usabilidade. Os aparelhos de televisão dos telespectadores não falham, o controle remoto sempre funciona (a menos que a bateria acabe) e, ao contrário da internet, a imagem não carrega mais lentamente nos horários de pico de visualização. Qualquer serviço de televisão interativa que cause problemas com o aparelho de televisão ou faça coisas inesperadas provavelmente não será tolerado pelos telespectadores.*

Este capítulo abordou os pontos-chave para construção de aplicações interativas na TV Digital brasileira, bem como, mostrar as oportunidades para o profissional de software nesse contexto. É possível afirmar que este pode ser visto como um novo nicho de atuação para o profissional de software e suas competências especialmente nas iniciativas de modelos de processo, ferramentas e técnicas de desenvolvimento de software.

3.8. Referências

ABNT NBR 15606-2 (2007) "Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações". *NBR 15606-2.*

ABNT NBR 15606-4 (2010) "Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 4: Ginga-J - Ambiente para a execução de aplicações procedurais". *NBR 15606-4*.

ABNT NBR 15606-5 (2008) "Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiodifusão digital Parte 5: Ginga-NCL para receptores portáteis – Linguagem de aplicação XML para codificação de aplicações". *NBR 15606-5.*

ARIB STD-B23 (2006) “Application Execution Engine Platform for Digital Broadcasting”. *ARIB Standard B23*.

CDC 1.1 (2008), “*Connected Device Configuration 1.1 - JSR 218*”. Sun Microsystems, disponível em http://jcp.org/en/jsr/detail?id=218

CESAR P; GEERTS D.; CHORIANOPOULOS K. (2009). “*Social Interactive Television: Immersive Shared Experiences and Perspectives*”. IGI Global, 1 ed, 2009.

CHORIANOPOULOS, K. (2004). “Virtual Television Channels: Conceptual Model, User Interface Design and Affective Usability Evaluation”. *Phd Thesis*. Dep. of Management Science and Technology Athens, University of Economics and Business. 2004. p. 180.

CPqD. (2006). “Modelo de referência: Sistema Brasileiro de Televisão Digital Terrestre.

ETSI TS 102 819 V1.3.1 (2005) “Digital Video Broadcasting (DVB): Globally Executable MHP (GEM) Specification 1.0.2”. European Telecommunications Standards Institute, *TS 102 819 V1.3.1*

FP 1.1 (2008) “*Foundation Profile 1.1 - JSR 219*”. Sun Microsystems, disponível em http://jcp.org/en/jsr/detail?id=219

Gawlinski, M. (2003). “*Interactive television production*”. Oxford: Focal Press, 2003.

ISO/IEC 13818-6 (1998) “Information Technology – Generic coding of moving pictures and associated audio information – Part 6: Extensions for DSMCC”. *ISO/IEC 13818-6*

ITU-T Recommendation J.200 (2001) “Worldwide common core – Application environment for digital interactive television services”. *ITU-T J.200*.

JavaDTV API (2009) “*Java DTV API 1.3 Specification*”. Sun Microsystems, disponível em URL: http://java.sun.com/javame/technology/javatv/

JavaTV API (2008) “*Java TV Specification 1.1 - JSR 927*”. Sun Microsystems. Disponível em URL: http://jcp.org/en/jsr/detail?id=927

JCE 1.0.1 (2006) “*Java Cryptography Extension - Security Optional Package Specification v1.0.1 – JSR 219*”. Sun Microsystems. Disponível em URL: http://jcp.org/en/jsr/detail?id=219

JSSE 1.0.1 (2006) “*Java Secure Socket Extension – Security Optional Package Specification v1.0.1 – JSR 219*”. Sun Microsystems. Disponível em URL http://jcp.org/en/jsr/detail?id=219

Leite, L. E. C., et al. (2005). “FlexTV – Uma Proposta de Arquitetura de middleware para o Sistema Brasileiro de TV Digital”. *Revista de Engenharia de Computação e Sistemas Digitais*. 2005, Vol. 2, pp. 29-50.

Lu, K. Y. (2005). “Interaction Design Principles for Interactive Television”. *Msc. Thesis*, Master of Science in Information Design and Technology. Georgia Institute of Technology. 2005. 202p.

Marques Neto M. C.; Santos, C. A. S. (2008) "An event-based model for interactive live TV shows". *In*: Proc. 16th ACM international conference on Multimedia, pages 845–848, New York, NY, USA, 2008. ACM.

Marques Neto M. C.; Santos, C. A. S. (2009). "StoryToCode: Um Modelo baseado em componentes para especificação de aplicações de TV Digital e Interativa convergentes". *In*: XV WebMedia, 2009, Fortaleza. SBC, 2009. v. 1. p. 59-66.

PBP 1.1 (2008), "Personal Basis Profile 1.1 – JSR 217", disponível em *URL*: http://jcp.org/en/jsr/detail?id=217

Perovich, D.; Bastarrica, M. C.; Rojas, C. (2009). "Model-Driven approach to Software Architecture design". *In*: ICSE Workshop on Sharing and Reusing Architectural Knowledge (May 16 - 16, 2009). IEEE, Washington, DC, 1-8.

Rodrigues R. F.; Soares L. F. (2006) "Produção de conteúdo declarativo para TV Digital". In: SEMISH, 2006.

SATSA 1.0.1 (2007) "Security and Trust Services API for J2ME – JSR 177". Sun Microsystems, disponível em: http://jcp.org/en/jsr/detail?id=177

Schwalb, E. M. (2003). "*iTV Handbook: Technologies and Standards*". Prentice Hall PRT, 2003.

Silva, L. D. N. (2008) "Uma Proposta de API para Desenvolvimento de Aplicações Multiusuário e Multidispositivo para TV Digital Utilizando o middleware Ginga", dissertação de mestrado, Universidade Federal da Paraíba, 2008.

Silva, L. D. N., et al. (2007). "Suporte para desenvolvimento de aplicações multiusuário e multidispositivo para TV Digital com Ginga". *T&C Amazônia Magazine*. N. 12, 2007, pp. 75-84

SILVA, F. P. R. (2010) "Xtation: um ambiente de execução e teste de aplicações interativas para o middleware Openginga", *Dissertação de mestrado*, Universidade Federal da Paraíba, 2010.

Soares, L. F. G. (2006). "MAESTRO: The Declarative middleware Proposal for the SBTVD". *In*: Proc. 4th European Interactive TV Conference. 2006

Soares, L. F. G., et al. (2007) "Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System". *Journal of the Brazilian Computer Society*. 2007, Vol. v12, pp. 37-46.

Soares, L. F. G; Moreno, M. F.; Sant'anna, F. F. G. (2009) "Ginga-NCL: Suporte a Múltiplos Dispositivos". *In*: Simpósio Brasileiro de Sistemas Multimídia e Hipermídia, 2009, Fortaleza. p. 43-50.

Souza Filho, G. L. et al. (2007) "Ginga-J: The Procedural middleware for the Brazilian Digital TV System". *Journal of the Brazilian Computer Society*. 2007, Vol. v12, pp. 47-56.

Steinberg, D. et al. (2009) "*EMF: Eclipse Modeling Framework 2.0*". 2nd ed. Addison-Wesley Professional.

Veiga, E. G.; Tavares, T. A. (2007). "Um Modelo de Processo para o Desenvolvimento de Programas para TV Digital e Interativa baseado em Metodologias Ágeis". *In*: 1o. Workshop em Desenvolvimento Rápido de Aplicações, 2007.

Whitaker, J. 2001. "*Interactive Television Demystified*", McGraw-Hill, 2001.

Yang, X.; Gu, P.; Dai, H. (2009). "Mapping Approach for Model Transformation of MDA Based on XMI/XML Platform". *In*: First Int. Work. on Education Technology and Computer Science – V.2. ETCS. IEEE, Washington, DC, Mar. 2009. 1016-1019.

Capítulo 4

Gerenciamento de Dados em Nuvem: Conceitos, Sistemas e Desafios

Flávio R. C. Sousa, Leonardo O. Moreira, José Antônio F. de Macêdo e Javam C. Machado

Abstract

Infrastructures, platforms and software are being offered as services, in cloud computing environments, with companies and users being able to rent computing power and storage in a transparent way and on-demand. These companies and users are moving their data and applications to the cloud so that they can access them anytime and anywhere. However, this new computing model requires significant changes in data management systems, since these systems demand scalability, availability, performance and cost. This mini-course will introduce the main concepts and technologies in cloud computing, focusing on data management and systems, and point research challenges and opportunities in this context.

Resumo

Infra-estruturas, plataformas e software estão sendo disponibilizados como serviços, sendo estes fornecidos por ambientes de Computação em Nuvem, onde empresas e usuários podem alugar capacidade de computação e armazenamento de forma transparente e sob demanda. Estas empresas e usuários estão movendo seus dados e aplicações para a nuvem de forma a acessá-los a qualquer momento e independente de localização. Entretanto, este novo modelo de computação requer grandes mudanças nos sistemas de gerenciamento de dados, pois estes sistemas necessitam de escalabilidade, disponibilidade, desempenho e custo. Este minicurso tem como objetivo apresentar os principais conceitos e tecnologias de computação em nuvem, destacando o gerenciamento de dados e sistemas, além de desafios e oportunidades de pesquisa neste contexto.

4.1. Introdução

Com o avanço da sociedade humana moderna, serviços básicos e essenciais são quase todos entregues de uma forma completamente transparente. Serviços de utilidade pública como água, eletricidade, telefone e gás tornaram-se fundamentais para nossa vida diária e são explorados por meio do modelo de pagamento baseado no uso [Vecchiola et al. 2009]. As infra-estruturas existentes permitem entregar tais serviços em qualquer lugar e a qualquer hora, de forma que possamos simplesmente acender a luz, abrir a torneira ou usar o fogão. O uso desses serviços é, então, cobrado de acordo com as diferentes políticas de tarifação para o usuário final. Recentemente, a mesma ideia de utilidade tem sido aplicada no contexto da informática e uma mudança consistente neste sentido tem sido feita com a disseminação de *Cloud Computing* ou Computação em Nuvem.

Computação em nuvem é uma tendência recente de tecnologia cujo objetivo é proporcionar serviços de Tecnologia da Informação (TI) sob demanda com pagamento baseado no uso. Tendências anteriores à computação em nuvem foram limitadas a uma determinada classe de usuários ou focadas em tornar disponível uma demanda específica de recursos de TI, principalmente de informática [Buyya et al. 2009]. Computação em nuvem pretende ser global e prover serviços para as massas que vão desde o usuário final que hospeda seus documentos pessoais na Internet até empresas que terceirizam toda infra-estrutura de TI para outras empresas. Nunca uma abordagem para a utilização real foi tão global e completa: não apenas recursos de computação e armazenamento são entregues sob demanda, mas toda a pilha de computação pode ser aproveitada na nuvem.

A computação em nuvem surge da necessidade de construir infra-estruturas de TI complexas, onde os usuários têm que realizar instalação, configuração e atualização de sistemas de software. Em geral, os recursos de computação e hardware são propensos a ficarem obsoletos rapidamente e a utilização de plataformas computacionais de terceiros é uma solução inteligente para os usuários lidarem com infra-estrutura de TI. Na computação em nuvem os recursos de TI são fornecidos como um serviço, permitindo que os usuários o acessem sem a necessidade de conhecimento sobre a tecnologia utilizada. Assim, os usuários e as empresas passaram a acessar os serviços sob demanda e independente de localização, o que aumentou a quantidade de serviços disponíveis.

Sistemas de gerenciamento de banco de dados são candidatos potenciais para a implantação em nuvem. Isso ocorre porque, em geral, as instalações destes são complexas e envolvem uma grande quantidade de dados, ocasionando um custo elevado, tanto em hardware quanto em software. Para muitas empresas, especialmente para *start-ups* e médias empresas, o pagamento baseado no uso do modelo de computação em nuvem, juntamente com o suporte para manutenção do hardware é muito atraente [Abadi 2009].

Embora os serviços de gerenciamento de banco de dados em nuvem tenham reduzido o custo de armazenamento de dados e melhorado o acesso, existe uma enorme complexidade envolvida na garantia de serviços de dados que podem escalar quando é necessário garantir operações consistentes e confiáveis considerando cargas de trabalho máximas. Além disso, o ambiente em nuvem tem requisitos técnicos para controlar centro de dados virtualizados, reduzindo os custos e aumentando a confiabilidade por meio da consolidação de sistemas em nuvem. Desafios tais como consistência e segurança dos dados são importantes para a computação em nuvem e acredita-se que futuras aplicações

centradas em dados irão alavancar os serviços de dados em nuvem.

Esta seção apresenta as definições, as características essenciais da computação em nuvem e os modelos de serviço e implantação. A seção 4.2 caracteriza o gerenciamento de dados em nuvem e a seção 4.3 apresenta os principais sistemas de gerenciamento de dados em nuvem. A seção 4.4 discute alguns desafios do gerenciamento de dados em nuvem. Finalmente, a seção 4.5 apresenta as conclusões finais.

4.1.1. Computação em Nuvem

A computação em nuvem está se tornando uma das palavras chaves da indústria de TI. A nuvem é uma metáfora para a Internet ou infra-estrutura de comunicação entre os componentes arquiteturais, baseada em uma abstração que oculta à complexidade de infra-estrutura. Cada parte desta infra-estrutura é provida como um serviço e, estes são normalmente alocados em centros de dados, utilizando hardware compartilhado para computação e armazenamento [Buyya et al. 2009].

A infra-estrutura do ambiente de computação em nuvem normalmente é composta por um grande número, centenas ou milhares de máquinas físicas ou nós físicos de baixo custo, conectadas por meio de uma rede como ilustra a Figura 4.1. Cada máquina física tem as mesmas configurações de software, mas pode ter variação na capacidade de hardware em termos de CPU, memória e armazenamento em disco [Soror et al. 2010]. Dentro de cada máquina física existe um número variável de máquinas virtuais (VM) ou nós virtuais em execução, de acordo com a capacidade do hardware disponível na máquina física.

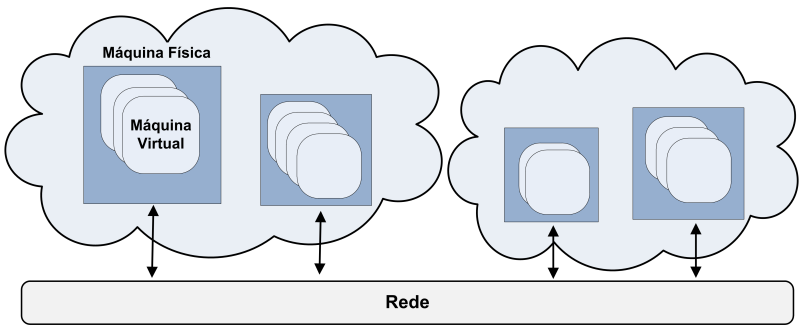


Figura 4.1. Ambiente de Computação em Nuvem

A computação em nuvem é uma evolução dos serviços e produtos de tecnologia da informação sob demanda, também chamada de *Utility Computing* [Brantner et al. 2008]. O objetivo da *Utility Computing* é fornecer componentes básicos como armazenamento, processamento e largura de banda de uma rede como uma "mercadoria" através de provedores especializados com um baixo custo por unidade utilizada. Usuários de serviços baseados em *Utility Computing* não precisam se preocupar com escalabilidade, pois a capacidade de armazenamento fornecida é praticamente infinita. A *Utility Computing* propõe fornecer disponibilidade total, isto é, os usuários podem ler e gravar dados a qualquer tempo, sem nunca serem bloqueados; os tempos de resposta são quase constantes e não dependem do número de usuários simultâneos, do tamanho do banco de dados ou de qualquer parâmetro do sistema. Os usuários não precisam se preocupar com *backups*,

pois se os componentes falharem, o provedor é responsável por substituí-los e tornar os dados disponíveis em tempo hábil por meio de réplicas [Brantner et al. 2008].

Uma razão importante para a construção de novos serviços baseados em *Utility Computing* é que provedores de serviços que utilizam serviços de terceiros pagam apenas pelos recursos que recebem, ou seja, pagam pelo uso. Não são necessários investimentos iniciais em TI e o custo cresce de forma linear e previsível com o uso. Dependendo do modelo do negócio, é possível que o provedor de serviços repasse o custo de armazenagem, computação e de rede para os usuários finais, já que é realizada a contabilização do uso.

Existem diversas propostas para definir o paradigma da computação em nuvem [Vaquero et al. 2009]. O *National Institute of Standards and Technology* (NIST) argumenta que a computação em nuvem é um paradigma em evolução e apresenta a seguinte definição: *"Computação em nuvem é um modelo que possibilita acesso, de modo conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente adquiridos e liberados com mínimo esforço gerencial ou interação com o provedor de serviços"* [Mell and Grance 2009]. NIST define que a computação em nuvem é composta por cinco características essenciais, três modelos de serviço e quatro modelos de implantação. Essas características e os modelos de computação em nuvem são detalhados a seguir.

4.1.1.1. Características Essenciais

As características essenciais são vantagens que as soluções de computação em nuvem oferecem. Algumas destas características, em conjunto, definem exclusivamente a computação em nuvem e fazem a distinção com outros paradigmas. Por exemplo, a elasticidade rápida de recursos, amplo acesso e a medição de serviço são características básicas para compor uma solução de computação em nuvem.

- *Self-service sob demanda*: O usuário pode adquirir unilateralmente recurso computacional, como tempo de processamento no servidor ou armazenamento na rede, na medida em que necessite e sem precisar de interação humana com os provedores de cada serviço.

- *Amplo acesso*: Recursos são disponibilizados por meio da rede e acessados através de mecanismos padronizados que possibilitam o uso por plataformas do tipo *thin*, tais como celulares, *laptops* e PDAs.

- *Pooling de recursos*: Os recursos computacionais do provedor são organizados em um *pool* para servir múltiplos usuários usando um modelo *multi-tenant* ou multi-inquilino, com diferentes recursos físicos e virtuais, dinamicamente atribuídos e ajustados de acordo com a demanda dos usuários. Estes usuários não precisam ter conhecimento da localização física dos recursos computacionais, podendo somente especificar a localização em um nível mais alto de abstração, tais como o país, estado ou centro de dados.

- *Elasticidade rápida*: Recursos podem ser adquiridos de forma rápida e elástica, em alguns casos automaticamente, caso haja a necessidade de escalar com o aumento da demanda, e liberados, na retração dessa demanda. Para os usuários, os recursos disponíveis para uso parecem ser ilimitados e podem ser adquiridos em qualquer quantidade e a qualquer momento.

- *Serviço medido*: Sistemas em nuvem automaticamente controlam e otimizam o uso de recursos por meio de uma capacidade de medição. A automação é realizada em algum nível de abstração apropriado para o tipo de serviço, tais como armazenamento, processamento, largura de banda e contas de usuário ativas. O uso de recursos pode ser monitorado e controlado, possibilitando transparência para o provedor e o usuário do serviço utilizado.

4.1.1.2. Modelos de Serviços

O ambiente de computação em nuvem é composto de três modelos de serviços. Estes modelos são importantes, pois eles definem um padrão arquitetural para soluções de computação em nuvem.

- *Software como um Serviço (SaaS)*: O modelo de SaaS proporciona sistemas de software com propósitos específicos que são disponíveis para os usuários por meio da Internet e acessíveis a partir de vários dispositivos do usuário por meio de uma interface *thin client* como um navegador Web. No SaaS, o usuário não administra ou controla a infra-estrutura subjacente, incluindo rede, servidores, sistema operacional, armazenamento, ou mesmo as características individuais da aplicação, exceto configurações específicas. Como exemplos de SaaS podemos destacar os serviços de *Customer Relationship Management* (CRM) da Salesforce e o Google Docs.

- *Plataforma como um Serviço (PaaS)*: O modelo de PaaS fornece sistema operacional, linguagens de programação e ambientes de desenvolvimento para as aplicações, auxiliando a implementação de sistemas de software. Assim como no SaaS, o usuário não administra ou controla a infra-estrutura subjacente, mas tem controle sobre as aplicações implantadas e, possivelmente, as configurações de aplicações hospedadas nesta infra-estrutura. *Google App Engine* e *Microsoft Azure* são exemplos de PaaS.

- *Infra-estrutura como um Serviço (IaaS)*: A IaaS torna mais fácil e acessível o fornecimento de recursos, tais como servidores, rede, armazenamento e outros recursos de computação fundamentais para construir um ambiente de aplicação sob demanda, que podem incluir sistemas operacionais e aplicativos. Em geral, o usuário não administra ou controla a infra-estrutura da nuvem, mas tem controle sobre os sistemas operacionais, armazenamento, aplicativos implantados e, eventualmente, seleciona componentes de rede, tais como *firewalls*. O Amazon *Elastic Cloud Computing* (EC2) e o Eucalyptus são exemplos de IaaS.

4.1.1.3. Modelos de Implantação

Quanto ao acesso e à disponibilidade, há diferentes tipos de modelos de implantação para o ambientes de computação em nuvem. A restrição ou abertura de acesso depende do processo de negócios, do tipo de informação e do nível de visão desejado.

- *Nuvem privada*: a infra-estrutura de nuvem é utilizada exclusivamente por uma organização, sendo esta nuvem local ou remota e administrada pela própria empresa ou por terceiros.
- *Nuvem pública*: a infra-estrutura de nuvem é disponibilizada para o público em geral, sendo acessado por qualquer usuário que conheça a localização do serviço.
- *Nuvem comunidade*: fornece uma infra-estrutura compartilhada por uma comunidade de organizações com interesses em comum.
- *Nuvem híbrida*: a infra-estrutura é uma composição de duas ou mais nuvens, que podem ser do tipo privada, pública ou comunidade e que continuam a ser entidades únicas, mas conectadas por meio de tecnologia proprietária ou padronizada que permite a portabilidade de dados e aplicações.

4.2. Gerenciamento de Dados em Nuvem

Sistemas de banco de dados em nuvem estão começando a ser utilizados e têm o potencial de atrair clientes de diversos setores do mercado, desde pequenas empresas com o objetivo de reduzir o custo total, por meio da utilização de infra-estrutura e sistemas de terceiros, até grandes empresas que buscam soluções que para gerenciar milhares de máquinas e permitir o atendimento de um aumento inesperado de tráfego.

A infra-estrutura de banco de dados em nuvem possui várias vantagens para os usuários: (i) previsibilidade e custos mais baixos, proporcional à qualidade do serviço (QoS) e cargas de trabalho reais, (ii) complexidade técnica reduzida, graças a *interfaces* de acesso unificado e a delegação de *tuning* e administração de banco de dados e (iii) a elasticidade e escalabilidade, proporcionando a percepção de recursos quase infinitos. Por outro lado, o provedor tem que garantir (i) a ilusão de recursos infinitos, sob cargas de trabalho dinâmicas e (ii) minimizar os custos operacionais associados a cada usuário [Abadi 2009].

Diversos sistemas e arquiteturas estão sendo desenvolvidos para suprir as novas demandas de aplicações com diferentes requisitos de processamento e armazenamento [Abouzeid et al. 2009]. Estes novos sistemas tentam fornecer uma visão de armazenamento e escalabilidade infinitos, mas tem que tratar o problema de provisionar recursos. Este problema, que em sistemas tradicionais consiste em determinar quais recursos são alocados para um único banco de dados, no ambiente em nuvem torna-se um problema de otimização, onde se tem uma grande quantidade de usuários, múltiplos sistemas de banco de dados e grandes centros de dados. Isso fornece uma oportunidade sem precedentes para explorar a economia em escala, balanceamento dinâmico de carga e gerenciamento de energia.

Esse aumento no número de abordagens disponíveis de gerenciamento de dados em nuvem agrava o problema da escolha, implantação e soluções de administração para o gerenciamento de dados. Com isso, os sistemas de banco de dados estão sendo disponibilizados como serviços, que encapsulam a complexidade do gerenciamento de dados por meio de formas de acesso simples e garantias de acordos de nível de serviço (SLA).

4.2.1. Requisitos

A definição dos requisitos é fundamental no gerenciamento de dados como um serviço. [Curino et al. 2010] apresentam uma lista de requisitos de um banco de dados como um serviço da perspectiva do usuário, do provedor e requisitos adicionais relacionados à nuvem pública conforme apresenta a Tabela 4.1.

Requisitos do Usuário	
U1	API simples com pouca configuração e administração (ex. sem tuning)
U2	Alto desempenho (ex. vazão, escalabilidade)
U3	Alta disponibilidade e confiança (ex. hot stand-by, backup)
U4	Acesso fácil à características avançadas (ex. snapshot, evolução de esquema, mineração de dados)
Requisitos do Provedor	
P1	Atender o SLA do usuário (ex. potencialmente sob carga de trabalho dinâmica)
P2	Limitar hardware e custo de energia (ex. multiplexação intensiva)
P3	Limitar custo de administração (ex. custo com pessoal)
Requisitos extra de Nuvem Pública	
P1	Esquema de preço: barato, previsível e proporcional ao uso (elasticidade)
P2	Garantias de segurança e privacidade
P3	Baixa latência (relevante para OLTP e aplicações Web)

Tabela 4.1. Requisitos para Banco de Dados como um Serviço [Curino et al. 2010]

Da perspectiva do usuário, a principal necessidade é um serviço de banco de dados com uma *interface* simples que não necessite de ajuste ou administração. Trata-se de uma melhoria em relação às soluções tradicionais que requerem soluções para provisionar recursos, seleção de sistemas de banco de dados, instalação, configuração e administração. O usuário quer um desempenho satisfatório, expresso em termos de latência e vazão, que é independente do tamanho da base de dados e das alterações da carga de trabalho. Atualmente, esta é uma tarefa difícil que exige uma ampla análise do pessoal de TI, software caro e atualizações de hardware. Alta disponibilidade é outro requisito fundamental, que normalmente é oferecido pelos sistemas de bancos de dados tradicionais, mas exige cuidados de configuração e manutenção. Finalmente, as características avançadas de gerenciamento do banco de dados, tais como *snapshot*, evolução de esquema e mineração de dados devem estar prontamente disponíveis e simples de utilizar.

Da perspectiva do provedor, é necessário atender aos acordos de nível de serviço, apesar da quantidade de dados e alterações na carga de trabalho. O sistema deve ser eficiente na utilização dos recursos de hardware. O modelo de serviço proporciona a oportunidade de fazer isso, por multiplexação de cargas de trabalho e ajuste dinâmico da alocação de recursos. Finalmente, a quantidade de tarefas de administração deve ser

minimizada. Para tanto, ferramentas sofisticadas de análise de carga de trabalho e centralização do gerenciamento de muitos bancos de dados devem ser utilizadas. Para provedores de serviços em nuvem pública, existem requisitos adicionais, tais como esquema de preço, segurança, privacidade e latência. Entretanto, estas questões não são específicas de bancos de dados e podem ser abordadas com técnicas em desenvolvimento pela comunidade de computação em nuvem.

4.2.2. Características

Alguns trabalhos destacam características específicas do gerenciamento de dados em nuvem. Segundo [Voicu and Schuldt 2009], no ambiente em nuvem, os dados são gerenciados em poucos centros de dados, utilizando recursos homogêneos e, mais recentemente, recursos heterogêneos. Estes dados são acessados por meio de APIs simples, SQL ou variações. Os sistemas de gerenciamento de dados em nuvem devem suportar atualizações concorrentes e transações ACID ou variações. Em relação à replicação, esta deve ser transparente para os usuários e fornecer garantias de qualidade de serviço, obtidas pela criação de réplicas dos dados em um mesmo centro de dados ou em centros de dados diferentes, além de permitir granulosidade fina dos dados. Na distribuição de dados, os sistemas de gerenciamento de dados em nuvem podem apresentar um controle global central ou distribuído, devem fornecer escalabilidade e suportar cargas de trabalho inesperadas. A Tabela 4.2 resume estas características.

Distribuição	Poucos centros de dados
Ambiente	Recursos homogêneos em centros de dados
Operações para acesso aos dados	API simples, SQL ou variações
Atualização	Suporte às atualizações concorrentes
Transações	ACID ou variações
Replicação	Garantia de QoS e transparência
Granulosidade da Replicação	Fina
Controle Global	Central ou Distribuído
Alterações Dinâmicas	Escalabilidade e suporte para cargas de trabalho inesperadas

Tabela 4.2. Características do Gerenciamento de Dados em Nuvem [Voicu and Schuldt 2009]

Uma característica essencial no ambiente de nuvem é o gerenciamento autônomo [Paton et al. 2009]. Hardware e software dentro de nuvens podem ser automaticamente reconfigurados, orquestrados e estas modificações são apresentadas ao usuário como uma imagem única. É possível identificar três fatores em comparação com os sistemas tradicionais: intervenção humana limitada, alta alternância na carga de trabalho e uma variedade de infra-estruturas compartilhadas [Agrawal et al. 2008]. Na maioria dos casos, não haverá administradores de sistemas de banco de dados ou de sistemas para ajudar os desenvolvedores que acessam uma base de dados em nuvem, fazendo com que a plataforma seja automatizada ao máximo. Os usuários podem variar a carga de trabalho habitual, necessitando de uma infra-estrutura de eficaz, pois esta será compartilhada por vários usuários ou inquilinos.

De forma a possibilitar a consolidação da computação em nuvem e dos sistemas de gerenciamento de banco de dados, técnicas de virtualização tem se tornando populares para a implantação destes sistemas e de outros sistemas de software [Soror et al. 2010]. A virtualização apoia a consolidação dos recursos nas empresas, pois permite que uma variedade de aplicações que funcionam com recursos de computação dedicados sejam movidos para um *pool* de recursos compartilhados, o que ajuda a melhorar a utilização dos recursos físicos, simplificar a administração de recursos e reduzir custos para a empresa. Em [Soror et al. 2010] é apresentado um estudo sobre a sobrecarga de executar sistemas de banco de dados em ambientes com máquinas virtuais. De acordo com o estudo, a execução não traz um alto custo de desempenho, visto que a virtualização introduz pouca sobrecarga para chamadas de sistema, tratamento de falta de páginas e operação de E/S no disco e isto não é traduzido em alta sobrecarga no tempo de execução da consulta. Chamadas de sistema e falta de páginas representam uma pequena fração no tempo de execução de uma consulta. Operações de E/S no disco correspondem a uma fração significativa do tempo, mas o retardo não é muito. Os resultados apresentados mostram que a sobrecarga média é menor do que 10%.

Associado à virtualização, o conceito de multi-inquilino é um ponto fundamental no gerenciamento de sistemas de banco de dados. Banco de dados multi-inquilino tem sido utilizado para hospedar múltiplos inquilinos dentro de um único sistema, permitindo o compartilhamento eficaz de recursos em diferentes níveis de abstração e isolamento. Existem vários modelos de multi-inquilino e estes podem compartilhar desde máquinas até tabelas. Por exemplo, a empresa Salesforce utiliza o modelo de tabela compartilhada [Weissman and Bobrowski 2009] enquanto [Soror et al. 2010] utilizam o modelo de máquina virtual compartilhada para melhorar a utilização de recursos. Algumas características do gerenciamento de dados em computação em nuvem aumentam a relevância de outros modelos de banco de dados multi-inquilino. Para melhorar a compreensão destes modelos, [Reinwald 2010] propõe um nova classificação, como mostra a Tabela 4.3.

Modo de Compartilhamento	**Isolamento**	**IaaS**	**PaaS**	**SaaS**
1. Hardware	VM	*x*		
2. Máquina Virtual	Usuário SO		*x*	
3. Sistema Operacional	Instância do BD		*x*	
4. Instância	BD		*x*	
5. Banco de Dados	Esquema		*x*	
6. Tabela	Linha			*x*

Tabela 4.3. Modelos de banco de dados multi-inquilino e correspondência com a computação em nuvem [Reinwald 2010] [Elmore et al. 2010]

Os modelos correspondentes às linhas 1-3 compartilham recursos nos níveis das mesmas máquinas físicas com diferentes níveis de abstração, por exemplo, múltiplas VMs, contas de usuários diferentes e diferentes instâncias do banco de dados. Neste caso não existe compartilhamento de recursos de banco de dados e as instâncias do banco de dados se mantêm independentes. As linhas 4-6 envolvem o compartilhamento de processos de banco de dados em vários níveis de isolamento, tais como diferentes banco de

dados, esquema ou *tablespace* e linha. Nos diferentes modelos, os dados dos inquilinos são armazenados de várias formas. O modelo de hardware compartilhado utiliza a virtualização para chavear várias VMs na mesma máquina. Cada VM possui apenas um processo de banco de dados e uma VM inteira, em geral, corresponde a um inquilino. Já o modelo de tabela compartilhada armazena dados de vários inquilinos em uma mesma tabela e algumas linhas de uma tabela correspondem a um inquilino.

4.2.2.1. Armazenamento e Processamento de Consultas

O armazenamento e processamento de consultas são pontos críticos no contexto da gestão de dados em nuvem [Armbrust et al. 2009]. Existem diversas abordagens para gerenciar dados em nuvem e cada sistema utiliza uma abordagem específica para persistir os dados. Dentre estas abordagens, podemos destacar novos sistemas de arquivos, *frameworks* e propostas para o armazenamento e processamento de dados. *Google File System* (GFS) [Ghemawat et al. 2003] é um sistema de arquivos distribuídos proprietário desenvolvido pelo Google e projetado especialmente para fornecer acesso eficiente e confiável aos dados usando grandes *clusters* de servidores. Em comparação com os sistemas de arquivos tradicionais, o GFS foi projetado e otimizado para ser executado em centros de processamento de dados e fornecer elevada vazão, baixa latência e tolerância a falhas individual de servidores. Inspirado pelo GFS, o projeto de código livre *Hadoop File System* (HDFS) [Hadoop 2010] armazena grandes arquivos em várias servidores e obtém a confiabilidade por meio da replicação de dados. Similar ao GFS, os dados são armazenados em nós geograficamente distribuídos.

Para apoiar o processamento distribuído de grandes conjuntos de dados em *clusters* foi introduzido pelo Google o *framework* MapReduce [Dean and Ghemawat 2004]. No modelo MapReduce cada operação é composta por duas funções: *Map* e *Reduce*. A função *Map* recebe uma porção do arquivo de entrada e, de acordo com a especificação do usuário, emite um conjunto de tuplas intermediárias no formato chave-valor. A função *Reduce* recebe um conjunto de valores associados a cada chave, chamados de blocos. O processamento, definido pelo usuário, é realizado sobre cada bloco. Por fim, cada função *Reduce* emite um conjunto de tuplas que são armazenadas em arquivos de saída. Existem algumas implementações de código livre, dentre as quais se destaca o *Hadoop MapReduce*.

Algumas propostas para o armazenamento e processamento utilizam a estrutura chave-valor em uma *Distributed Hash Table* (DHT) [DeCandia et al. 2007]. Este valor e a chave associada são armazenados de forma desnormalizada, o que facilita a distribuição dos dados entre os nós do sistema, onde cada um destes nós possui parte dos dados. As APIs básicas de acesso são simples com operações tais como *get(key)* e *put(key, value)*. APIs mais sofisticadas permitem a execução de funções definidas pelo usuário no ambiente do servidor tais como *execute(key, operation, parameters)* ou *mapreduce(keyList, mapFunc, reduceFunc)*. Para acessar ou modificar qualquer dado é necessário fornecer uma chave, já que a busca é realizada utilizando a igualdade. É possível realizar pesquisas com outros critérios, tais como "maior ou menor" ou expressões booleanas. Neste caso são utilizadas técnicas de busca exaustiva e árvores B+ distribuídas.

Outra abordagem para armazenar e processar dados em nuvem consiste em utilizar uma estrutura de colunas ou *arrays* multidimensionais [Chang et al. 2006]. Os dados são organizados em tabelas e estas possuem diversas colunas. Cada coluna armazena um valor, acessado por meio de uma chave. Nesta abordagem, todos os valores de uma coluna são serializados em conjunto, os valores da coluna seguinte também, e assim por diante. Com isso, os dados semelhantes, de mesmo formato, podem ser agrupados, auxiliando no armazenamento e na recuperação de informação. Além disso, diversas técnicas tais como a fragmentação de dados e o processamento OLAP tornam-se mais eficientes para dados gerenciados com esta abordagem. Outras abordagens para o armazenamento e processamento de consultas gerenciam os dados de tal sorte a refletir a estrutura de um documento ou tratam os dados na forma de grafos.

4.2.2.2. Transações

Os conceitos de processamento de transações são de extrema importância em ambientes centralizados e distribuídos, sendo que nestes últimos é comum haver dados replicados e alocados em locais geograficamente distantes. Estes conceitos definem o nível de consistência e integridade dos dados nestes ambientes. A utilização de transações distribuídas define o controle do processamento de dados em todo ambiente de computação em nuvem e tem a responsabilidade de garantir as propriedades ACID ou variações destas no ambiente. Neste sentido, necessita-se utilizar protocolos de replicação de dados, terminação distribuída e sincronização de acesso devido à natureza compartilhada dos recursos.

Um ponto fundamental na construção de sistemas distribuídos e considerado por todos os sistemas em nuvem é o teorema *Consistency*, *Availability*, *Partition Tolerance* (CAP) [Brewer 2000] [Gilbert and Lynch 2002]. Este teorema mostra que os sistemas distribuídos não podem assegurar as seguintes propriedades simultaneamente:

- *Consistência*: cada cliente tem sempre a mesma visão dos dados.
- *Disponibilidade*: todos os clientes podem sempre ler e escrever.
- *Tolerância a partições*: o sistema funciona corretamente em várias partições de rede física.

Um sistema distribuído pode suportar apenas duas dessas três propriedades. Dessa forma, os sistemas em nuvem que utilizam estratégias baseadas em fragmentação dos dados são forcados a escolher entre consistência e disponibilidade [Brantner et al. 2008]. Em decorrência deste teorema, algumas abordagens para o gerenciamento de dados em nuvem têm utilizado diferentes formas de consistência. Uma alternativa é utilizar a abordagem *Basically Available, Soft state, Eventually consistent* (BASE) [Pritchett 2008], que é caracterizada pelo sistema ser basicamente disponível, pois este parece estar em funcionamento todo o tempo; em estado leve, já que o sistema não precisa estar sempre consistente; e eventualmente consistente, que define que o sistema torna-se consistente em um determinado momento.

De acordo com [Vogels 2009], existem duas formas de caracterizar a consistência: *do ponto de vista do programador/cliente* - como os dados são observados e atualizados

e *do ponto de vista do servidor* - como é processado o fluxo das atualizações por meio do sistema e que garantias este pode fornecer, no que diz respeito às atualizações. Por exemplo, podem-se definir alguns tipos de consistência. A *consistência forte* garante que após uma atualização ser concluída, qualquer acesso subsequente fornecerá o valor atualizado. Por outro lado, na *consistência fraca*, o sistema não garante que os acessos subsequentes irão retornar o valor atualizado. O período entre uma atualização e o momento no qual é garantido ao observador que este irá sempre visualizar o valor atualizado é denominado "janela de inconsistência".

A *consistência eventual* é uma forma específica de consistência fraca, na qual o sistema de banco de dados garante que, se nenhuma atualização for feita ao dado, todos os acessos irão devolver o último valor atualizado. Se não ocorrer nenhum erro, o tamanho máximo da "janela de inconsistência" pode ser determinado com base em fatores tais como os atrasos de comunicação, a carga do sistema e o número de réplicas envolvidas do esquema de replicação. O sistema mais popular que implementa consistência eventual é o *Domain Name System* (DNS).

O modelo de consistência eventual apresenta um número de variações tais como consistência causal, consistência leitura/escrita, consistência de sessão, consistência de leitura monótona e consistência de escrita monótona. Estas variações podem ser combinadas com o objetivo de tornar mais simples a construção de aplicações e permitir aos sistemas de banco de dados melhorarem a consistência e fornecer maior disponibilidade. Sob o ponto de vista do servidor, o nível de consistência depende de como as atualizações são propagadas entre as réplicas de dados. Isto permite melhorar a taxa de transferência e fornecer escalabilidade. Contudo, o desenvolvedor deve estar consciente sobre quais garantias de consistência são fornecidas pelo sistema de banco de dados na construção de aplicações.

4.2.2.3. Escalabilidade e Desempenho

A nuvem é composta por uma enorme rede de máquinas que necessita ser escalável. A escalabilidade deve ser transparente para os usuários, podendo estes armazenar seus dados na nuvem sem a necessidade de saber a localização dos dados ou a forma de acesso. Na nuvem, os desenvolvedores dispõem de ambientes escaláveis, mas eles têm que aceitar algumas restrições sobre o tipo de software que se pode desenvolver, desde limitações que o ambiente impõe na concepção das aplicações até a utilização de sistemas de banco de dados do tipo chave-valor, ao invés de sistemas de banco de dados relacionais.

De forma geral, é possível identificar duas dimensões de escalabilidade: vertical e horizontal. Na escalabilidade vertical melhora-se a capacidade do hardware, incrementando individualmente os nós existentes, como por exemplo, por meio da disponibilização de um servidor com mais memória física ou da melhoria da largura de banda que conecta dois nós. Isto funciona razoavelmente bem para os dados, mas tem várias limitações tais como a aquisição constante de hardware de maior capacidade, o que pode criar dependência de fornecedores, acrescendo os custos.

A escalabilidade horizontal consiste em adicionar mais máquinas à solução atual de tal modo que seja possível distribuir as requisições entre estas máquinas. No caso

dos dados, pode-se agrupá-los por funções e distribuí-los por vários bancos de dados. Dessa forma, ocorre a fragmentação dos dados em sistemas de bancos de dados e cada um destes sistemas pode ser dimensionado de forma independente. Este tipo de escalabilidade oferece maior flexibilidade, mas necessita de um planejamento específico.

A escalabilidade envolvendo dados é uma tarefa complexa, já que a maioria dos sistemas de bancos de dados utiliza arquiteturas não compartilhadas, tornando a disposição dos dados um ponto chave. Pode-se pensar que adicionar dinamicamente um novo servidor de banco de dados é tão simples como dividir os dados em mais um servidor. Por exemplo, se há dois servidores, cada um com 50% do total dos dados e se adiciona um terceiro servidor, poder-se-ia colocar um terço dos dados em cada servidor, fazendo com que cada um dos três servidores ficasse com 33% dos dados. Entretanto, não é tão simples assim, pois as consultas dos usuários envolvem dados relacionados em servidores diferentes, o que exige o transporte dos dados, diminuindo o desempenho do sistema. Por esta razão, a fragmentação dos dados deve ser feito de forma a minimizar o transporte de dados, o qual adiciona um custo relevante ao processamento.

Em relação ao desempenho, as soluções de gerenciamento de dados em nuvem devem lidar com problemas de tempo de resposta, em virtude das diferentes tecnologias e heterogeneidade do hardware utilizado, o que pode influenciar o desempenho. Por exemplo, uma falha de hardware, tais como problemas no acesso ao núcleo de uma máquina com múltiplos *cores* ou a disputa por recursos não virtualizados, causa degradação do desempenho de uma máquina do sistema. Se a carga de trabalho necessária para executar uma consulta é dividida igualmente entre as máquinas com configurações diferentes, ocasiona um atraso no processamento, visto que o tempo para completar a consulta será aproximadamente igual ao tempo para a execução da máquina com menor configuração para completar a tarefa atribuída.

4.2.2.4. Tolerância a Falhas e Distribuição de Dados

Diferentemente das abordagens anteriores, onde se procurou evitar falhas por meio da utilização de hardware de custo elevado, as infra-estruturas para nuvem são construídas em cima de hardware de baixo custo e com a suposição de que máquinas e redes podem falhar. Dessa forma, as soluções desenvolvidas devem lidar com falhas, já que estas irão ocorrer em algum momento [Abadi 2009]. Para tratar as falhas, as soluções em nuvem utilizam técnicas que auxiliam a distribuição dos dados, tais como DHT, que facilita a fragmentação dos dados. Por meio da fragmentação dos dados é possível melhorar a disponibilidade e distribuir a carga, tanto para operações de escrita quanto de leitura. Se apenas uma máquina falhar, os dados pertencentes a essa máquina são afetados, mas não o armazenamento de dados como um todo.

Em relação ao processo de replicação, pode-se criar e iniciar novas réplicas rapidamente por meio de máquinas virtuais [Soror et al. 2010]. Isso permite manter muitas réplicas por máquina, o que reduz a quantidade total de armazenamento e, portanto, os custos associados. O processo de replicação pode ser realizado de forma síncrona ou assíncrona ou uma combinação destas. Isso determina o nível de consistência, confiabilidade e desempenho. Dentre os protocolos utilizados na replicação de dados pode-se destacar o

de cópia primária, réplica ativa, quorum baseado em 2PC, *Paxos* [Chang et al. 2006] e o *Gossip* [DeCandia et al. 2007].

4.2.3. Classificação

Existem diversos sistemas para o gerenciamento de dados em nuvem, cada um destes com características e propósitos específicos. Com o objetivo de auxiliar o estudo destes sistemas e realizar um comparativo entre os mesmos, propomos uma classificação com base nos seguintes parâmetros: *modelo relacional* e *nativo para nuvem*. O primeiro parâmetro se refere à utilização do modelo relacional pelo sistema e o segundo parâmetro está relacionado ao processo de construção do sistema, ou seja, se o sistema foi concebido para atender as necessidades da computação em nuvem. Os sistemas com modelos de dados em comum foram agrupados de forma a facilitar a compreensão. Vale ressaltar que alguns destes sistemas possuem características de mais de um modelo de dados. A Figura 4.2 apresenta essa classificação e destaca alguns sistemas.

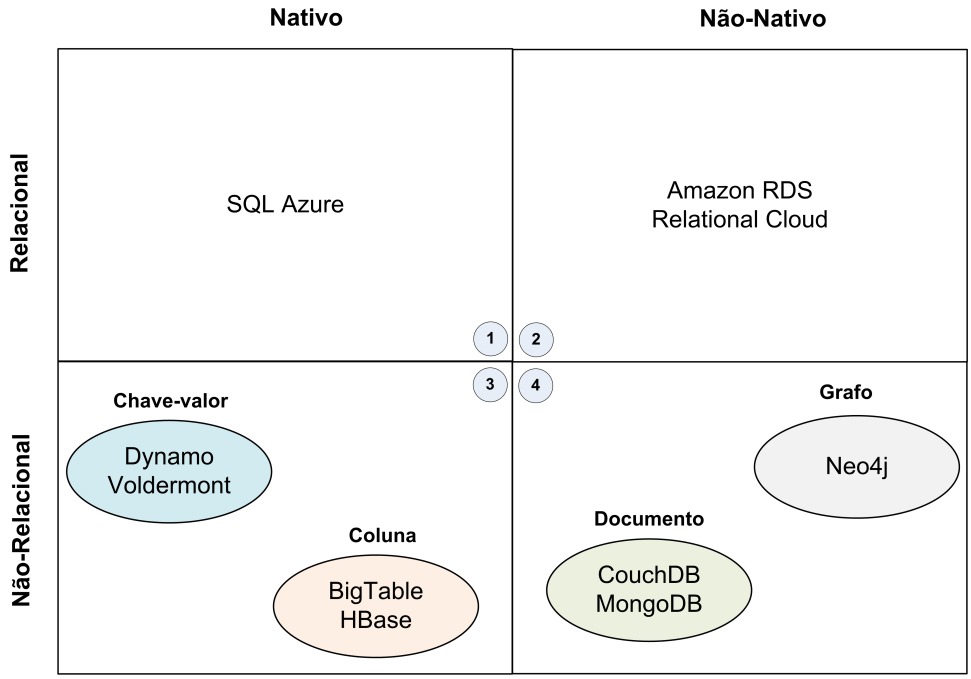


Figura 4.2. Classificação dos Sistemas de Gerenciamento de Dados em Nuvem

No primeiro quadrante estão os sistemas de banco de dados relacionais desenvolvidos nativamente para nuvem. Estes sistemas estão sendo concebidos considerando as características da computação em nuvem juntamente com aspectos do modelo relacional. O exemplo mais conhecido destes sistemas é o Microsoft SQL Azure. No segundo quadrante estão os sistemas de banco de dados relacionais tradicionais que não foram concebidos para nuvem, mas que podem ser executados em uma infra-estrutura baseada em nuvem. Como exemplo destes sistemas destacam-se o Amazon *Relational Database Service* (Amazon RDS) [Amazon 2010] e o Relational Cloud [Curino et al. 2010].

No terceiro quadrante estão os sistemas considerados nativos para nuvem que não utilizam o modelo relacional, tais como os sistemas que utilizam o modelo chave-valor

e baseado em coluna. Dentre os sistemas que utilizam o modelo chave-valor pode-se destacar o Amazon Dynamo [DeCandia et al. 2007] e o Voldemort. BigTable, HBase e Cassandra são exemplos de sistemas baseados em coluna. No quarto quadrante estão os sistemas não-nativos que não utilizam o modelo relacional, tais como grafo, documento ou XML. Estes sistemas estão sendo desenvolvidos para propósitos específicos e utilizados em ambientes em nuvem. O Neo4j é um exemplo dos sistemas baseados em grafo e CouchDB e MongoDB são exemplos de sistemas de banco de dados orientados a documentos.

Os sistemas não-relacionais, coletivamente, iniciaram um movimento denominado *Not only SQL* (NoSQL). NoSQL é um termo genérico para uma classe definida de sistemas de banco de dados não-relacionais que foram projetados para gerenciar grandes volumes de dados e, em geral, fornecem garantias de consistência fraca, como consistência eventual e utilizam estruturas e *interfaces* simples. Estes sistemas estão sendo utilizando principalmente em serviços de hospedagem de sítios, redes sociais e em outros serviços específicos.

4.3. Sistemas para o Gerenciamento de Dados em Nuvem

Existe uma grande quantidade de sistemas para o gerenciamento de dados em nuvem e assim, não é possível destacar cada um destes sistemas, tais como o Amazon RDS, Noe4j [Neo 2010], Yahoo PNUTS [Cooper et al. 2008], entre outros [Agrawal et al. 2010]. A seguir destacamos alguns destes sistemas e apresentamos uma análise comparativa considerando diversas características do gerenciamento de dados.

4.3.1. Microsoft SQL Azure

O Microsoft SQL Azure é composto por um conjunto de serviços para o armazenamento e processamento de dados em nuvem [Azure 2010]. O SQL Azure é parte do Windows Azure Storage, sendo que o Azure Storage também inclui armazenamento persistente por meio de *blobs*, *tables* e filas. Um *blob* é um par nome, objeto que permite armazenar objetos de até 50 GB. *Tables* são diferentes das tabelas relacionais e são compostas de entidades. Elas não são acessadas usando a linguagem SQL, mas por serviços de dados. Já as filas fornecem um serviço de troca de mensagens persistentes e confiável.

O *SQL Azure Database* (SAD) é o principal componente do SQL Azure e foi construído com base na tecnologia do sistema de banco de dados relacional SQL Server. Para permitir uma integração transparente de aplicações com SQL Azure, o SAD suporta os principais comandos da linguagem *Transact-SQL* (T-SQL). Esta linguagem possui diversas características, dentre os quais podemos destacar tabelas, índices, funções, procedimentos e gatilhos, entre outros. O gerenciamento do SQL Azure pode ser realizado por meio da ferramenta denominada *Houston*.

O SQL Azure implementa alta disponibilidade, tolerância a falhas e o conceito de multi-inquilino. Cada banco de dados é implementado como uma partição de dados replicados em múltiplas máquinas em um *SQL Azure Datacenter*. Com esse tipo de abordagem, SQL Azure fornece um gerenciamento automático de falhas e balanceamento de carga de trabalho. A estratégia de replicação utiliza três cópias dos itens de dados para fornecer a disponibilidade e implementa consistência forte. No *SQL Azure Database*, um

banco de dados individual possui um tamanho limitado a 50 GB. Para criar soluções que armazenem dados maiores do que este tamanho deve-se fragmentar grandes conjuntos de dados entre múltiplos bancos de dados e utilizar consultas em paralelo para acessá-los. Entretanto, o particionamento dos dados é realizado de forma manual.

4.3.2. Amazon S3/SimpleDB

O Amazon S3 é um sistema de armazenamento distribuído desenvolvido com base no Dynamo [DeCandia et al. 2007]. O Dynamo utiliza o modelo par chave-valor armazenados em uma DHT e não possui suporte a associações ou esquemas. Para garantir um nível de escalabilidade e disponibilidade, de acordo com o teorema CAP, o Dynamo relaxa a consistência em alguns cenários de falhas e faz uso de versões de objetos e resolução de conflitos assistida por meio de uma *interface* para os desenvolvedores. No Dynamo, as operações de leitura e escrita utilizam identificadores únicos, sendo realizadas sobre objetos binários de até 5GB e não existe suporte às operações sobre múltiplos objetos. As propriedades das transações ACID apresentam as seguintes características: atomicidade e isolamento são garantidos pela escrita de um único objeto; durabilidade é obtida por meio de escrita replicada e apenas a consistência não é forte. A API do Dynamo possui as operações *get()* e *put()*.

Escalabilidade, balanceamento de carga e suporte à heterogeneidade foram aspectos enfatizados na concepção do Dynamo e tornam relativamente simples adicionar nós, distribuir as requisições e suportar nós com características diferentes. As propriedades de simetria e descentralização são utilizadas de tal forma que todos os pontos são igualmente responsáveis e com o objetivo de evitar pontos únicos de falhas. O Dynamo usa a técnica de *hashing consistente* para prover o particionamento e a replicação. Para tratar o problema de balanceamento de carga, decorrente da utilização de poucos nós ou nós heterogêneos, o Dynamo utiliza a estratégia de "nós virtuais", onde cada nó físico pode ter vários nós virtuais associados. Dessa forma, máquinas com maior capacidade de processamento e armazenamento podem ter uma maior quantidade de nós, sendo estes distribuídos aleatoriamente. O Dynamo utiliza o protocolo *Gossip* para verificar se um nó pertence ao sistema e para detecção de falhas.

O Amazon S3 utiliza o conceito de objeto, entidade fundamental que consiste de dados, metadados e um identificador único. Os objetos são organizados em *buckets* e estes servem para agrupar objetos ou espaço de nomes. Para tratar a ocorrência de falhas, os dados são propagados entre os centros de dados de forma postergada e o usuário pode especificar a localização geográfica de um *buckets*. No S3 as operações de escrita são atômicas, mas podem ser executadas sobre múltiplas chaves. Entretanto, este sistema não fornece mecanismos de bloqueio.

O SimpleDB é um sistema de armazenamento hierárquico de dados construído para superar as limitações do S3. O SimpleDB possui atributos múltiplos, indexação e suporte a consultas. O armazenamento de dados é livre de esquema, escalável e eficiente para cenários onde as operações de leituras são predominantes, tais como fóruns de discussão e *backup*. O modelo de dados do SimpleDB utiliza o conceito de domínios, que podem ser comparados a uma tabela em sistemas de banco de dados relacionais, com a diferença que um domínio pode conter um conjunto diferente de atributos para cada

item. Cada item pode ter vários valores para cada atributo. Os itens são identificados por domínio e os atributos são par chave-valor, sem definição de tipo, ou seja, uma cadeia de caracteres indexados automaticamente. Para criar relacionamentos no SimpleDB, o desenvolvedor pode desnormalizar seus dados ou tratar as relações na camada de aplicação. O SimpleDB possui um linguagem de consultas semelhante ao SQL e suporta seleção, operadores, ordenação e contagens, mas uma consulta pode ser processada apenas em um domínio. O SimpleDB tem algumas restrições em relação ao tamanho e número do domínios, tempo da consulta, não permite junções e comparações de itens dentro de uma consulta, suporte a transações, entre outros.

4.3.3. Bigtable/Google App Engine datastore

O BigTable é um sistema de armazenamento de dados distribuído em larga escala. Ele funciona como um sistema de banco de dados orientado a colunas e utiliza o GFS para gerenciar os dados, podendo ser utilizado juntamente com o *MapReduce* para distribuir o processamento dos dados [Chang et al. 2006]. O BigTable não suporta um modelo relacional, mas fornece um modelo simples e flexível. No modelo de dados do BigTable, uma tabela é um mapa esparso, distribuído, persistente e multidimensional, organizado em três dimensões: linha, coluna e hora, formando uma célula. Linhas com chaves consecutivas são agrupadas em *tablets*, que são unidades de distribuição e balanceamento de carga. As linhas são a unidade básica de atomicidade e alterações de uma única linha são sempre transacionais. As colunas são agrupadas em famílias que formam a unidade de controle. Várias versões de uma mesma célula podem ser armazenadas, indexadas pela hora. BigTable não suporta nenhum tipo de junção e relações são gerenciadas na camada de aplicação.

O Bigtable tem três componentes principais: uma biblioteca comum aos clientes, um servidor mestre e vários servidores de *tablets*. O servidor mestre é responsável por designar *tablets* para os servidores de *tablets*, balanceamento de cargas e por gerenciar alterações nos esquemas. Cada servidor de *tablet* responde às requisições de leituras e escritas nas tabelas e o serviço de bloqueio denominado *Chubby* é usado para administrar os servidores. O serviço *Chubby* utiliza o protocolo baseado em quorum *Paxos* para resolver o problema de consenso distribuído e fornecer bloqueios exclusivos para os arquivos gerenciados. O BigTable fornecer consistência forte, mas não replica o banco de dados diretamente, pois um *tablet* pode ser atribuído para apenas um servidor de *tablet* por vez. A replicação dos dados é realizada por meio do GFS, que gerencia o armazenamento dos *tablets* e dos arquivos de *log*.

O Google App Engine (GAE) datastore é uma solução de gerenciamento de dados baseado no BigTable. O GAE *datastore* utiliza um modelo de dados livre de esquema, suporta tipos de dados primitivos e armazena os objetos de forma serializada no BigTable. O GAE *datastore* suporta transações realizadas sobre várias entidades, mas exige que todos os objetos em uma transação estejam dentro do mesmo grupo de entidade. O GAE *datastore* utiliza consistência forte, similar ao BigTable. Muitos tipos de dados comuns são suportados no armazenamento de dados, incluindo *String*, *int*, *float*, e *DateTime*. O GAE *datastore* utiliza uma chave como identificador único para atributos como linha, links, entre outros e pode ser acessado com a linguagem de consulta *Google Query Language* (GQL), um subconjunto da linguagem SQL. A linguagem GQL possui suporte a

seleção, ordenação e alguns operadores. A instrução de seleção pode ser realizada em uma única tabela, ou seja, não suporta junções, assim como associações ou chaves compostas em decorrência do modelo de dados utilizado. Dessa forma, associações devem ser implementadas na camada de aplicação.

4.3.4. CouchDB

O CouchDB é um sistema de banco de dados orientado a documento e livre de esquema [Apache 2010]. O CouchDB possui uma série de características que torna sua utilização viável em servidores que possuem hardware de baixo desempenho e utiliza técnicas de armazenamento e controle de concorrência baseadas na estrutura do documento. Este sistema foi implementado utilizando a plataforma *Erlang OTP* devido à ênfase desta plataforma em tolerância a falhas e oferece escalabilidade, disponibilidade e confiabilidade, mesmo ao executar em hardware que está sujeito à falhas.

O CouchDB organizada os dados em documentos e cada um destes possui um identificador único. Cada documento pode ter qualquer quantidade de atributos e cada atributo pode conter listas ou objetos. Os documentos são armazenados e acessados como objetos *JavaScript Object Notation* (JSON). Operações de atualização são executadas sobre todo o documento e o CouchDB gerencia as alterações por meio de um identificador de revisão contido em cada documento. O CouchDB utiliza estrutura de arquivo baseada em árvores B+ para persistir os dados.

O CouchDB não utiliza os conceitos de tabela, chave, relacionamento ou junção. O modelo de consulta do CouchDB consiste nos conceitos de *visões*, que são construídas utilizando funções *MapReduce* e de uma API de consultas via *http*. Uma *visão* é basicamente uma coleção de pares chave-valor que, juntamente com o *MapReduce* permite ao usuário criar um programa de mapeamento e outro de redução para filtrar e agregar dados em uma base de documentos. O CouchDB implementa o protocolo de controle de concorrência multi-versão (MVCC) adaptado para documento e utiliza o modelo de consistência eventual. Adicionalmente, o CouchDB possui um mecanismo incremental de replicação com detecção e gerenciamento bi-direcional de conflitos.

4.3.5. Cassandra

Cassandra é um sistema de armazenamento distribuído para o gerenciamento de grandes quantidades de dados espalhados por centenas de máquinas [Lakshman and Malik 2010]. Este sistema foi desenvolvido para ter uma alta disponibilidade, consistência eventual, escalabilidade incremental, replicação otimista e com baixo custo de administração. O sistema Cassandra pode funcionar em hardware de baixo custo e lida com alta taxa de escrita sem sacrificar a eficiência na leitura. Por utilizar uma arquitetura distribuída, Cassandra não possui um ponto único de falha. Este sistema aumenta a tolerância às falhas, e, com isso, aumenta a confiabilidade do sistema, já que os dados são automaticamente replicados em vários nós de um ou mais centros de dados e o protocolo *Gossip* é utilizado para tratar às falhas. A vazão de operações de leitura e escrita no sistema pode crescer linearmente e novas máquinas são adicionadas sem nenhum custo ou interrupção para as aplicações.

Cassandra é um híbrido que combina a arquitetura descentralizada do Dynamo

com o modelo de dados do BigTable e possui uma API simples como meio de acesso aos dados. Cassandra possui um modelo de dados na forma de um *hash* de quatro ou cinco dimensões: espaço de chave, família de colunas, coluna, chave e super-coluna. O espaço de chave é um local para as famílias de colunas e, tipicamente, é definida uma por aplicação. Uma família de colunas é o local para agrupar as colunas, assim como uma tabela em um sistema de banco de dados relacional e é acessada por meio de uma chave. Uma chave é similar aos outros sistemas de armazenamento chave-valor. Uma super-coluna pode ser comparada a coluna que possui sub-colunas e é utilizada para modelar tipos de dados complexos.

A coluna é a menor unidade de dados existente no modelo de dados do Cassandra e é formada por uma tripla que contém um nome, um valor e um *timestamp*. Todos os valores são fornecidos pelo usuário, incluindo o *timestamp*. Neste caso, os relógios nos usuários devem ser sincronizados, pois estes *timestamps* são usados para resolução de conflitos. Uma família de colunas contém uma lista ordenada de colunas, que o usuário pode fazer referência ao nome da coluna. Cada família de coluna é armazenada em um arquivo separado e este arquivo é classificado por uma chave, que determina qual dado é armazenado na máquina. Cassandra determina os nós responsáveis pelos dados e, quando um usuário submete uma solicitação de escrita para um nó aleatório, as operações de escrita são registradas e então aplicadas a uma versão na máquina. O *log de consolidações* é armazenado em um disco dedicado para a máquina. Para operações de escrita não existem bloqueios e a atomicidade é garantida pelo acesso por meio de uma chave. Além disso, as escritas são aceitas durante cenários de falhas, o que aumenta a disponibilidade.

Cassandra não fornece índices automaticamente e o usuário precisa definir soluções para melhorar o desempenho. Uma alternativa é utilizar estratégias que envolvam família de colunas por consulta. Com isso, é possível melhorar a organização dos dados e minimizar o esforço no acesso aos dados e no processamento de consultas. Cassandra também possui algumas limitações, como por exemplo, todos os dados de um único registro devem pertencer ao disco de uma máquina do cluster. Isso porque as chaves dos registros são utilizadas para determinar os nós responsáveis por replicar dados. Outra limitação é que o valor de uma coluna não pode ultrapassar 2GB.

4.3.6. Relational Cloud

Relational Cloud é um sistema de banco de dados como um serviço concebido com o objetivo de consolidar as funcionalidades de gestão de dados para grandes empresas e *outsourcing* do gerenciamento de dados em nuvem para pequenas e médias empresas [Curino et al. 2010]. O Relational Cloud prove disponibilidade por meio de replicação transparente, gerenciamento automático das cargas de trabalho e migração de dados em tempo de execução, além de oferecer suporte a transações distribuídas serializáveis.

O Relational Cloud foca na fragmentação dos dados, migração de dados em tempo de execução e alocação e análise de carga de trabalho. Em relação à fragmentação, o Relational Cloud propõe um novo algoritmo para a fragmentação de base dados com o objetivo de minimizar a probabilidade de uma determinada operação ter que acessar múltiplos nós para fornecer uma resposta. A migração em tempo de execução é obtida por meio de uma estratégia que prevê quando uma adaptação será necessária antes que algum nó do

sistema seja sobrecarregado. Para tanto, técnicas de fragmentação e movimentação de pequenos conjuntos de dados são realizadas durante a execução. A alocação e análise de carga de trabalho são realizadas por meio de técnicas estáticas e dinâmicas de caracterização da carga de trabalho, seleção de sistemas de armazenamento, atribuição de cargas de trabalho para as instâncias de banco de dados e atribuição de instâncias de banco de dados para nós físicos.

O Relational Cloud foi projetado para ser executado em máquinas em um único centro de dados. Cada máquina física executa várias instâncias de sistema de banco de dados. Estas instâncias podem utilizar sistemas de armazenamento diferentes, visto que *engines* especializados, em geral, são eficientes para tarefas específicas. Cada banco de dados é dividido em partições lógicas por meio de técnicas de fragmentação. Essas partições são armazenadas em grupos de réplicas com o objetivo de garantir a disponibilidade e tolerância a falhas. Um grupo de réplica consiste de várias instâncias do banco de dados sendo que cada uma armazena cópia dos dados de uma partição lógica. A fragmentação dos dados e alocação dos grupos de réplicas nas máquinas são controladas pelo analisador de carga de trabalho.

A comunicação entre as aplicações e o Relational Cloud é realizada por meio de *interfaces* padrão ou protocolos conhecidos, tais como a *interface* JDBC. As consultas SQL recebidas são enviadas para um roteador, responsável por analisar e verificar os metadados do banco de dados e determinar o plano de execução. Por fim, o sistema de transações distribuídas distribui a carga de trabalho, assegurando a serializabilidade e o tratamento de falhas. Por meio do monitoramento constante de desempenho e carga de trabalho, o Relational Cloud ajusta a fragmentação dos dados e as opções de localização em tempo de execução. Falhas no sistema e alterações de carga de trabalho exigem a evolução do esquema de fragmentação e alocação em tempo de execução. Para tanto, é necessária a migração dos dados baseada nas instâncias do mecanismo de armazenamento, o que melhora o desempenho do sistema.

4.3.7. Análise Comparativa

Novos serviços de banco de dados em nuvem, como o GAE datastore, Amazon S3, SimpleDB e Cassandra utilizam modelos simplificados de dados baseados em par chave-valor ou orientados a coluna. Os dados são armazenados em estruturas otimizadas e acessados por meio de APIs simples. GAE datastore, S3/SimpleDB, CouchDB não possuem suporte a transações ACID. Cassandra utiliza um conceito simplificado de transações para linhas [Candan et al. 2009]. Estes sistemas suportam apenas consistência eventual, exceto o GAE datastore e todos apresentam alta escalabilidade e alta disponibilidade, menos o CouchDB que possui média escalabilidade, em decorrência do modelo de dados utilizado.

GAE datastore, Amazon S3, SimpleDB e Cassandra apresentam bons resultados na manipulação de grandes volumes de dados, pois os modelos de dados utilizados por estes sistemas favorecem a fragmentação dos dados. Contudo, em geral, estes sistemas não suportam operações como junções, possuem apenas garantias de consistência fraca e fornecem suporte limitado a transações. Além disso, utilizando um modelo de dados mais simples, a complexidade é empurrada para as demais camadas da aplicação, exigindo dos desenvolvedores a adição de funcionalidades mais complexas nas camadas superiores.

CouchDB e sistemas baseados em grafo optaram por modelos mais ricos de dados. Com isso, eles apresentam abstrações mais poderosas que tornam mais fácil modelar domínios complexos. Estes modelos de dados mais ricos introduzem maior quantidade de ligação entre os dados, o que prejudicar a escalabilidade. Vale ressaltar que estes sistemas ainda são muito recentes e existem poucos estudos detalhados sobre os mesmos, assim como ferramentas e tecnologias de apoio para sua ampla utilização.

Característica	S3 SimpleDB	BigTable GAE	Cassandra	CouchDB	SQL Azure	Relational Cloud
Modelo	Chave-valor	Coluna	Coluna	Documento	Relacional	Relacional
Armazenamento	Hash consistente	Índices	Índices	Árvore B+	Tabela	Tabela
Linguagem de Consulta	API simples	API simples	API simples	API simples	SQL	SQL
Transações	Não	Não	Sim simplificada	Não	Sim	Sim
Consistência	Eventual	Forte	Eventual	Eventual	Forte	Forte
Escalabilidade	Alta	Alta	Alta	Média	Média	Baixa
Disponibilidade	Alta	Alta	Alta	Alta	Alta	Média

Tabela 4.4. Comparativo entre os Sistemas de Ger. de Dados em Nuvem

Os sistemas SQL Azure e Relational Cloud utilizam o modelo de dados relacional, que fornece grande flexibilidade no acesso aos dados, tais como agregação e consultas com junções. Eles implementam transações ACID e garantia de consistência forte, o que facilita no desenvolvimento de aplicações. Em relação à escalabilidade e disponibilidade, SQL Azure e Relational Cloud apresentam características diferentes. O SQL Azure possui média escalabilidade, já que não oferece suporte a transações distribuídas ou consultas através de múltiplas partições de dados localizados em diferentes nós, mas apresenta alta disponibilidade. O sistema Relational Cloud apresenta baixa escalabilidade e média disponibilidade, pois se trata de um sistema em desenvolvimento e que não considera aspectos de elasticidade e ambientes virtualizados, características essenciais da computação em nuvem. O SQL Azure não possui suporte a fragmentação automática de dados, essencial para melhorar a escalabilidade, mas o Relational Cloud apresenta iniciativas neste sentido. A Tabela 4.4 mostra um resumo deste comparativo.

É importante ressaltar que os modelos de dados utilizados pelos sistemas de gerenciamento de dados em nuvem são compatíveis, ou seja, pode-se expressar um conjunto de dados em qualquer um deles. Por exemplo, é possível decompor todos os dados de um banco de dados relacional em uma coleção de par chave-valor. Dessa forma, existem contextos onde cada um destes modelos de dados é mais apropriado. De acordo com [Kossmann et al. 2010], os principais provedores tem adotado diferentes arquiteturas para seus serviços e o custo e o desempenho deste serviços variam significativamente dependendo da carga de trabalho.

4.4. Desafios e Oportunidades de Pesquisa

A computação em nuvem apresenta diversas vantagens, mas também possui uma série de desafios que devem ser superados para sua ampla utilização. A seguir destacamos alguns destes desafios no contexto do gerenciamento de dados. Outros desafios envolvendo aspectos gerais da computação em nuvem podem ser encontrados em [Armbrust et al. 2009] [Sousa et al. 2009] [Zhang et al. 2010] .

4.4.1. Armazenamento e Processamento de Consultas

Com o aumento no volume de dados e dos requisitos para extrair valores a partir destes dados, empresas necessitam gerenciar e analisar uma grande quantidade de dados e fornecer alto desempenho. Além disso, os dados são gerenciados em várias partições, o que dificulta garantias transacionais, tais como atomicidade e isolamento. Para tratar estes problemas, diferentes soluções tem sido desenvolvidas combinando tecnologias como o MapReduce e sistemas de banco de dados paralelos [Abouzeid et al. 2009]. Um desafio é definir uma arquitetura para paralelizar o processamento de consultas com ênfase em consultas *On Line Analytical Processing* (OLAP) expressas em SQL ou em novas linguagens [Olston et al. 2008]. Esta arquitetura deve focar na interação entre o mecanismo de processamento de consultas e os sistemas de arquivos paralelos, tais como o HDFS e proporcionar diferentes níveis isolamento.

Frameworks como MapReduce e suas diversas implementações como o Hadoop e Drayd foram projetados para o processamento distribuído de tarefas e geralmente utilizam sistemas de arquivos como o GFS e o HDFS. Estes sistemas de arquivos são diferentes dos sistemas de arquivos distribuídos tradicionais em sua estrutura de armazenamento, padrão de acesso e *interface* de programação. Em particular, eles não implementam a *interface* padrão POSIX, e, portanto, apresentam problemas de compatibilidade com aplicações e sistemas de arquivos legados [Zhang et al. 2010]. Um ponto importante é desenvolver soluções para tratar a compatibilidade com os sistemas de arquivos distribuídos tradicionais, tais como métodos para suportar o *framework* MapReduce usando sistemas de arquivos para cluster e técnicas para o acesso concorrente aos dados.

Em relação ao acesso aos serviços de dados em nuvem, os provedores fornecem linguagens com restrições, APIs simples e estas apresentam muitas limitações, tais como a ausência de operações de junções. Considerando a grande quantidade de serviços de dados em nuvem, os desenvolvedores necessitam utilizar APIs diferentes para cada serviço. Isso exige mais esforço dos desenvolvedores e aumenta a complexidade na camada de aplicação. Um desafio é fornecer um API comum para vários serviços, assim como uma linguagem de consulta robusta e com diversas características dos sistemas tradicionais [Cooper et al. 2009]. Outros aspectos relevantes neste contexto estão relacionados a otimização e *tuning* de sistemas de banco de dados em nuvem [Agrawal et al. 2008].

4.4.2. Escalabilidade e Consistência

As soluções em nuvem focam em escalabilidade, mas sacrificam a consistência. Estas soluções permitem apenas consultas por uma chave, não suportam múltiplas chaves ou consultas com junções e oferecem consistência fraca de dados, por exemplo, consistência eventual. Este tipo de consistência não permite a construção de uma ampla gama de

aplicações, tais como serviços de pagamento e leilões *online*, que não podem trabalhar com dados inconsistentes [Wei et al. 2009]. Neste contexto, aspectos de armazenamento de dados, processamento de consultas e controle transacional têm sido flexibilizados por algumas abordagens para garantir a escalabilidade, mas ainda não existem soluções que combinem estes aspectos de forma a melhorar o desempenho sem comprometer a consistência dos dados [Abadi 2009]. De acordo com [Armbrust et al. 2009], o desenvolvimento de um sistema de armazenamento que combine os diversos aspectos de computação em nuvem, de forma a aumentar a escalabilidade, a disponibilidade e consistência dos dados é um problema de pesquisa em aberto. Dessa forma, um desafio é desenvolver soluções escaláveis e com consistência forte.

Em [Wei et al. 2009] é apresentado uma solução com suporte a transações ACID, sem comprometer a escalabilidade mesmo na presença de falhas. Contudo, esta solução necessita utilizar técnicas de desnormalização de esquemas, o que pode dificultar sua utilização. [Yang et al. 2009] propõem uma plataforma para o gerenciamento de dados que pode escalar para uma grande quantidade de pequenas aplicações e fornecer consistência forte. Para tanto, várias técnicas de sistemas de banco de dados, tais como replicação, migração e gerenciamento de SLA para garantir vazão e disponibilidade foram desenvolvidas.

4.4.3. Gerenciamento de Banco de Dados Virtualizados

A tecnologia de virtualização tornou-se comum em modernos centros de dados e sistemas de *clusters* [Soror et al. 2010]. Enquanto muitos sistemas de banco de dados já estão sendo utilizados em ambientes virtualizados, questões relacionadas à utilização destes sistemas ainda permanecem em aberto [Rogers et al. 2010]. Uma vez que o uso de máquinas virtuais pode auxiliar serviços sob demanda, um desafio é explorar a possibilidade de escalar sistemas de banco de dados para tratar as cargas de trabalho inesperadas por meio da utilização de novas réplicas em máquinas virtuais recém provisionadas [Soror et al. 2010]. Com isso é necessário garantir o acesso consistente ao sistema banco de dados durante e após o processo de replicação, coordenar solicitações de roteamento para as máquinas virtuais antigas e novas, desenvolver políticas para a provisão de novas réplicas e modelos mais abrangentes para o planejamento da capacidade necessária para a nuvem [Aboulnaga et al. 2009]. Também é importante desenvolver soluções para otimizar as consultas em sistemas de banco de dados executados em ambientes virtualizados e ajustar os parâmetros do banco de dados com a máquina virtual visando melhorar a interação entre estes [Soror et al. 2010].

Outro ponto fundamental é a alocação de recursos, já que as máquinas físicas trabalham com apenas parte de sua capacidade. Estas máquinas poderiam ser melhor utilizadas, permitindo a redução de custos. Uma questão relevante é determinar a melhor alocação das aplicações para as máquinas virtuais de acordo com métricas que maximizem a utilização dos recursos. Em [Rogers et al. 2010] é proposto uma solução para a alocação das sistemas de banco de dados em ambientes virtualizados. Entretanto, não são abordados aspectos relacionados à replicação dos dados.

4.4.4. Qualidade do Serviço de Dados

Em ambientes de computação em nuvem, a qualidade de serviço é uma característica definida entre o provedor e o usuário e expressa por meio de SLA. Com isso, o usuário do serviço tem algumas garantias, tais como desempenho e disponibilidade. Apesar das limitações de rede e segurança, as soluções em nuvem devem fornecer elevado desempenho, além de serem flexíveis para se adaptar diante de uma determinada quantidade de requisições. Como, em geral, os ambientes de computação em nuvem possuem acesso público, torna-se imprevisível e variável a quantidade de requisições realizadas, dificultando fazer estimativas e fornecer garantias de QoS. As soluções em nuvem necessitam estimar métricas de qualidade tendo como base o estado global do sistema e o histórico de processamentos do mesmo. Entretanto, estimar métricas de qualidade é um fator desafiador nestes ambientes, pois os recursos gerenciados estão, freqüentemente, sendo expandidos ou realocados no intuito de melhorar o desempenho.

Uma questão relevante para garantir a qualidade em qualquer infra-estrutura compartilhada é isolar o desempenho de aplicações diferentes. Aplicações podem adicionar uma carga variável sobre a nuvem e é necessário verificar como esta carga de trabalho irá afetar as outras aplicações que compartilham o mesmo hardware. Algumas abordagens para este problema utilizam quotas de recursos ou compartilhamento ponderado [Cooper et al. 2009]. Independente da abordagem utilizada, esta terá que garantir a qualidade de serviço, mesmo em condições extremas e com diferentes componentes da nuvem. Também é importante garantir a qualidade do serviço de dados independente da forma de acesso. Por exemplo, o provedor deve fornecer um serviço com garantias para um usuário que acessa este serviço por meio de um desktop ou de um dispositivo móvel. Para tanto, é necessário identificar a requisição e, caso necessário, alocar mais recursos de forma a garantir o SLA com o usuário. Técnicas adaptativas e dinâmicas deverão ser desenvolvidas para tornar os sistemas de gerenciamento de dados em nuvem viáveis [Aboulnaga et al. 2009], além de ferramentas e processos que automatizem tarefas tais como a alocação de recursos [Agrawal et al. 2008].

4.4.5. Sistemas de Banco de Dados Multi-Inquilino

No sistema de banco de dados como um serviço, os inquilinos acessam o sistema e compartilham recursos, o que pode interferir no desempenho do sistema. Dessa forma, o provisionamento de recursos deve ser eficiente, já que as cargas de sistemas de banco de dados com um serviço são muito variáveis, visto que os inquilinos podem acessar o sistema com maior freqüência em determinados momentos. Com a ampla utilização de ambientes virtualizados, altamente compartilhados, a questão do provisionamento torna-se determinante. A distribuição dos dados é realizada por meio do particionamento e replicação e deve garantir que os dados dos inquilinos não sejam perdidos e que a recuperação seja simples. Para tratar a recuperação, pode-se estender a linguagem de consulta de forma a permitir o acesso considerando a distribuição dos dados dos inquilinos.

Muitas soluções para construir sistemas de banco de dados multi-inquilino necessitam de reescrita da consulta [Hui et al. 2009]. Isso ocasiona um esforço adicional para o sistema, o que pode comprometer a escalabilidade. Novas propostas tais como o BigTable permitem fornecer alternativas para a construção de sistemas de banco de dados

multi-inquilino e podem ser incorporadas aos sistemas existentes, melhorando o suporte multi-inquilino. Dessa forma, um desafio é desenvolver uma arquitetura multi-inquilino para sistemas de banco de dados e técnicas para maximizar a quantidade de inquilinos, com baixo custo e garantias de desempenho.

4.4.6. Segurança dos Dados

A computação em nuvem é um modelo que utiliza a Internet para disponibilizar seus serviços. Isso se torna mais complexo visto que os recursos computacionais utilizam diferentes domínios de redes, sistemas operacionais, software, criptografia, políticas de segurança, entre outros. Questões de segurança devem ser consideradas para prover a autenticidade, confidencialidade e integridade. No que diz respeito à confiabilidade e responsabilidade, o provedor deve fornecer recursos confiáveis, especialmente se a computação a ser realizada é crítica e deve existir uma delimitação de responsabilidade entre o provedor e o usuário. Dessa forma, devem-se ter meios para impedir o acesso não autorizado a informações e que os dados sensíveis permaneçam privados, pois estes podem ser processados fora das empresas [Agrawal et al. 2009]. Em geral, cada sistema tem seu próprio modelo de dados e política de privacidade destes dados [Cooper et al. 2009]. Quando ocorre a movimentação de dados entre sistemas, deve-se garantir a privacidade dos dados mesmo com mudança entre modelo de dados diferente e que aplicações multi-inquilino acessem dados de outras aplicações apenas de acordo com as políticas definidas.

Técnicas de criptografia podem ser utilizadas para garantir a privacidade dos dados. No entanto, estas técnicas têm implicações significativas de desempenho de consultas em sistemas de bancos de dados. Dessa forma, alternativas para a integração de técnicas de criptografia com sistemas de bancos de dados devem ser investigados e desenvolvidas, já que a complexidade computacional da criptografia de dados aumenta o tempo de resposta da consulta. Em [Agrawal et al. 2009] é apresentado uma abordagem segura e escalonável para preservar a privacidade. Em vez de utilizar a criptografia, que é computacionalmente caro, é utilizada uma estratégia de distribuição dos dados em vários sítios do provedor e técnicas para acessar as informações de forma secreta e compartilhada.

4.4.7. Descrição e Descoberta de Serviços de dados

Na computação em nuvem vários modelos evoluíram rapidamente para aproveitar as tecnologias de software, plataformas de programação, armazenamento de dados e infraestrutura de hardware como serviços [Youseff et al. 2008]. Enquanto estes modelos se referem ao núcleo dos serviços de computação em nuvem, suas inter-relações têm sido ambíguas e a viabilidade de sua interoperabilidade é questionável. Além disso, cada serviço da nuvem tem *interfaces* e protocolos diferentes e é complexo para os usuários encontrar e compor serviços, visto que os diversos serviços estão dispersos na Internet e possuem características distintas. Por exemplo, suponha que um usuário necessite de um serviço de processamento e outro de armazenamento para persistir os dados processados. Uma alternativa para o usuário seria fazer uma busca exaustiva. Contudo, como existe uma grande quantidade de serviços, isso pode se tornar inviável. Além disso, ainda seria necessário compor os serviços de processamento e armazenamento, o que seria outra dificuldade.

Dessa forma, um desafio é desenvolver técnicas eficazes para descrever, descobrir e compor serviços na nuvem de forma a auxiliar os usuários em suas tarefas. Ontologias podem ser utilizadas para a organização do domínio de conhecimento de computação em nuvem, seus componentes e suas relações, ajudando na descrição e descoberta de serviços em nuvem [Youseff et al. 2008], assim como na composição de novos serviços a partir dos serviços existentes. Isso ajudará no projeto de serviços com interoperabilidade entre diferentes provedores, proporcionando melhorias na qualidade dos serviços.

4.4.8. Avaliação de Serviços de Dados em Nuvem

A avaliação de serviços de dados em nuvem apresenta diferenças significativas em relação aos sistemas de banco de dados tradicionais. Sistemas tradicionais pressupõem a existência de configurações fixa de recursos, tratam exclusivamente da otimização de desempenho e tem como objetivo minimizarem o tempo de resposta para cada requisição. Essa visão não considera os custos operacionais do sistema. Entretanto, o modelo de pagamento baseado no uso da computação em nuvem requer que custos operacionais sejam considerados juntamente com o desempenho. No ambiente em nuvem, o objetivo é minimizar a quantidade de recursos necessários para garantir uma meta de tempo de resposta para cada requisição [Florescu and Kossmann 2009].

Para garantir a disponibilidade e a tolerância a falhas, os serviços de dados em nuvem fornecem diferentes garantias de consistência, tais como consistência eventual. Consistência forte implica em alto custo por transação e, em algumas situações, reduz a disponibilidade. Consistência fraca apresenta menor custo, mas resulta em alto custo operacional, por exemplo, *overselling* de um produto em uma loja virtual. Em [Kraska et al. 2009] é apresentado um novo paradigma que permite aos desenvolvedores definir garantias de consistência e o chaveamento automático destas garantias em tempo de execução com o objetivo de minimizar os custos.

Existem algumas iniciativas para a avaliação de serviços de dados em nuvem. [Florescu and Kossmann 2009] destacam que sistemas de *benchmark* para gerenciamento de dados devem tratar de aspectos de custo, tempo de resposta, vazão, escalabilidade, consistência, flexibilidade e discute como o novo problema de otimização de banco de dados pode impactar na arquitetura dos modernos sistemas de banco de dados. [Binnig et al. 2009] discutem porque *benchmark* tradicionais não são suficientes para analisar os novos serviços em nuvem e apresentam idéias para o desenvolvimento de um novo *benchmark* . Em [Cooper et al. 2010] é apresentado um *framewok* com o objetivo de facilitar a avaliação de diferentes sistemas de dados em nuvem. Este *framewok* trata características de desempenho e escalabilidade, mas ainda não aborda aspectos de disponibilidade e replicação.

4.5. Conclusão

A computação como um serviço está finalmente emergindo e as empresas podem prestar serviços diretamente aos usuários por meio da Internet de acordo com as suas necessidades. Neste contexto, a computação em nuvem é um paradigma que está cada vez mais popular. Diversas empresas apresentaram suas iniciativas na promoção da computação em nuvem. A comunidade científica também tem apresentado algumas iniciativas, principalmente com foco em suas necessidades. Este trabalho apresentou os principais aspectos

de computação em nuvem, destacando o gerenciamento de dados.

Foi possível perceber que a computação em nuvem ainda não tem uma definição clara e completa na literatura, mas existe um grande esforço neste sentido. No gerenciamento de dados, os sistemas de bancos de dados estão evoluindo rapidamente e os desenvolvedores podem escolher o modelo de dados mais adequado para suas aplicações. No futuro, as infra-estruturas serão compostas tanto por sistemas de banco de dados relacionais como também por sistemas baseados nos modelos chave-valor, coluna, documento, grafo, entre outros.

Por fim, foram discutidos alguns desafios importantes no gerenciamento de dados, tais como escalabilidade, segurança, qualidade do serviço de dados, entre outros. É importante ressaltar que, várias soluções, existentes em outros modelos computacionais, que solucionem ou atenuem estes desafios, podem ser aplicadas em ambientes de computação em nuvem. Estes desafios geram oportunidades de pesquisa que devem ser superados, de forma que computação em nuvem seja amplamente aceita e utilizada por todos.

Referências

[Abadi 2009] Abadi, D. J. (2009). Data management in the cloud: Limitations and opportunities. *IEEE Data Eng. Bull.*, 32:3–12.

[Aboulnaga et al. 2009] Aboulnaga, A., Salem, K., Soror, A. A., Minhas, U. F., Kokosielis, P., and Kamath, S. (2009). Deploying database appliances in the cloud. *IEEE Data Eng. Bull.*, 32(1):13–20.

[Abouzeid et al. 2009] Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D. J., Rasin, A., and Silberschatz, A. (2009). Hadoopdb: An architectural hybrid of mapreduce and dbms technologies for analytical workloads. *PVLDB*, 2(1):922–933.

[Agrawal et al. 2010] Agrawal, D., Abbadi, A. E., Antony, S., and Das, S. (2010). Data management challenges in cloud computing infrastructures. In *Databases in Networked Information Systems, 6th International Workshop, DNIS 2010*, volume 5999 of *Lecture Notes in Computer Science*, pages 1–10. Springer.

[Agrawal et al. 2009] Agrawal, D., Abbadi, A. E., Emekci, F., and Metwally, A. (2009). Database management as a service: Challenges and opportunities. *Data Engineering, International Conference on*, 0:1709–1716.

[Agrawal et al. 2008] Agrawal, R., Garcia-Molina, H., Gehrke, J., Gruenwald, L., Haas, L. M., Halevy, A. Y., Hellerstein, J. M., Ioannidis, Y. E., Korth, H. F., Kossmann, D., Madden, S., Ailamaki, A., Magoulas, R., Ooi, B. C., O'Reilly, T., Ramakrishnan, R., Sarawagi, S., Stonebraker, M., Szalay, A. S., Weikum, G., Bernstein, P. A., Brewer, E. A., Carey, M. J., Chaudhuri, S., Doan, A., Florescu, D., and Franklin, M. J. (2008). The claremont report on database research. *ACM SIGMOD Record*, 37:9.

[Amazon 2010] Amazon (2010). *Amazon Relational Database Service (Amazon RDS)*. http://aws.amazon.com/rds/.

[Apache 2010] Apache (2010). *Apache CouchDB*. http://couchdb.apache.org.

[Armbrust et al. 2009] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M. (2009). Above the

clouds: A berkeley view of cloud computing. Technical report, EECS Department, University of California, Berkeley.

[Azure 2010] Azure (2010). *Microsoft Azure*. http://www.microsoft.com/azure/.

[Binnig et al. 2009] Binnig, C., Kossmann, D., Kraska, T., and Loesing, S. (2009). How is the weather tomorrow?: towards a benchmark for the cloud. In *DBTest '09: Proceedings of the Second International Workshop on Testing Database Systems*, pages 1–6, New York, NY, USA. ACM.

[Brantner et al. 2008] Brantner, M., Florescu, D., Graf, D., Kossmann, D., and Kraska, T. (2008). Building a database on s3. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08*, page 251, New York. ACM Press.

[Brewer 2000] Brewer, E. A. (2000). Towards robust distributed systems (abstract). In *PODC*, page 7. ACM.

[Buyya et al. 2009] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616.

[Candan et al. 2009] Candan, K. S., Li, W.-S., Phan, T., and Zhou, M. (2009). Frontiers in information and software as services. In *ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 1761–1768, Washington, DC, USA. IEEE Computer Society.

[Chang et al. 2006] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2006). Bigtable: a distributed storage system for structured data. In *OSDI '06: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, pages 15–15, Berkeley, CA, USA. USENIX Association.

[Cooper et al. 2009] Cooper, B. F., Baldeschwieler, E., Fonseca, R., Kistler, J. J., Narayan, P. P. S., Neerdaels, C., Negrin, T., Ramakrishnan, R., Silberstein, A., Srivastava, U., and Stata, R. (2009). Building a cloud for yahoo! *IEEE Data Eng. Bull.*, 32(1):36–43.

[Cooper et al. 2008] Cooper, B. F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.-A., Puz, N., Weaver, D., and Yerneni, R. (2008). Pnuts: Yahoo!'s hosted data serving platform. *PVLDB*, 1(2):1277–1288.

[Cooper et al. 2010] Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2010). Benchmarking cloud serving systems with ycsb. In *SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154, New York, NY, USA. ACM.

[Curino et al. 2010] Curino, C., Jones, E., Zhang, Y., Wu, E., and Madden, S. (2010). Relational cloud: The case for a database service. Technical report, MIT-CSAIL-TR-2010-014. Computer Science and Artificial Intelligence Laboratory, MIT, USA.

[Dean and Ghemawat 2004] Dean, J. and Ghemawat, S. (2004). Mapreduce: simplified data processing on large clusters. In *OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*, pages 10–10, Berkeley, CA, USA. USENIX Association.

[DeCandia et al. 2007] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220.

[Elmore et al. 2010] Elmore, A., Das, S., Agrawal, D., and Abbadi, A. E. (2010). Who's driving this cloud? towards efficient migration for elastic and autonomic multitenant databases. Technical Report CS 2010-05, University of California, Santa Barbara, CA, USA.

[Florescu and Kossmann 2009] Florescu, D. and Kossmann, D. (2009). Rethinking cost and performance of database systems. *SIGMOD Rec.*, 38(1):43–48.

[Ghemawat et al. 2003] Ghemawat, S., Gobioff, H., and Leung, S.-T. (2003). The google file system. *SIGOPS Oper. Syst. Rev.*, 37(5):29–43.

[Gilbert and Lynch 2002] Gilbert, S. and Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59.

[Hadoop 2010] Hadoop (2010). *Apache Hadoop*. http://hadoop.apache.org.

[Hui et al. 2009] Hui, M., Jiang, D., Li, G., and Zhou, Y. (2009). Supporting database applications as a service. In *ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 832–843, Washington, DC, USA. IEEE Computer Society.

[Kossmann et al. 2010] Kossmann, D., Kraska, T., and Loesing, S. (2010). An evaluation of alternative architectures for transaction processing in the cloud. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*, pages 579–590, New York, NY, USA. ACM.

[Kraska et al. 2009] Kraska, T., Hentschel, M., Alonso, G., and Kossmann, D. (2009). Consistency rationing in the cloud: Pay only when it matters. *PVLDB*, 2(1):253–264.

[Lakshman and Malik 2010] Lakshman, A. and Malik, P. (2010). Cassandra: a decentralized structured storage system. *SIGOPS Oper. Syst. Rev.*, 44(2):35–40.

[Mell and Grance 2009] Mell, P. and Grance, T. (2009). *Draft NIST Working Definition of Cloud Computing*. National Institute of Standards and Technology. http://csrc.nist.gov/groups/SNS/cloud-computing.

[Neo 2010] Neo (2010). *Neo4j - Graph Database*. http://neo4j.org.

[Olston et al. 2008] Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. (2008). Pig latin: a not-so-foreign language for data processing. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110, New York, NY, USA. ACM.

[Paton et al. 2009] Paton, N. W., Aragão, M. A. T., Lee, K., Fernandes, A. A. A., and Sakellariou, R. (2009). Optimizing utility in cloud computing through autonomic workload execution. *IEEE Data Eng. Bull.*, 32(1):51–58.

[Pritchett 2008] Pritchett, D. (2008). Base: An acid alternative. *Queue*, 6(3):48–55.

[Reinwald 2010] Reinwald, B. (2010). Database support for multi-tenant applications. *In IEEE Workshop on Information and Software as Services (WISS). Co-located with ICDE*.

[Rogers et al. 2010] Rogers, J., Papaemmanouil, O., and Cetintemel, U. (2010). A generic auto-provisioning framework for cloud databases. *In Proceeding of the 5th International Workshop on Self-Managing Database Systems (SMDB). Co-located with ICDE*.

[Soror et al. 2010] Soror, A. A., Minhas, U. F., Aboulnaga, A., Salem, K., Kokosielis, P., and Kamath, S. (2010). Automatic virtual machine configuration for database workloads. *ACM Trans. Database Syst.*, 35(1):1–47.

[Sousa et al. 2009] Sousa, F. R. C., Moreira, L. O., and Machado, J. C. (2009). *Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios*. In: MOURA, R. S. (Org.) ; SOUZA, F. V. (Org.) ; OLIVEIRA, A. C. (Org.). Escola Regional de Computação (Ceará, Maranhão e Piauí, ERCEMAPI 2009, 1. ed. EDUFPI, Piauí.

[Vaquero et al. 2009] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2009). A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55.

[Vecchiola et al. 2009] Vecchiola, C., Chu, X., and Buyya, R. (2009). *Aneka: A Software Platform for .NET-based Cloud Computing*, pages 267–295. In: W. Gentzsch, L. Grandinetti, G. Joubert (Eds.). High Speed and Large Scale Scientific Computing. IOS Press, Amsterdam, Netherlands.

[Vogels 2009] Vogels, W. (2009). Eventually consistent. *Commun. ACM*, 52(1):40–44.

[Voicu and Schuldt 2009] Voicu, L. C. and Schuldt, H. (2009). How replicated data management in the cloud can benefit from a data grid protocol: the re:gridit approach. In *CloudDB '09: Proceedings of the First International Workshop on Cloud Data Management*, pages 45–48, New York, NY, USA. ACM.

[Wei et al. 2009] Wei, Z., Pierre, G., and Chi, C.-H. (2009). Scalable transactions for web applications in the cloud. In *Euro-Par*, pages 442–453.

[Weissman and Bobrowski 2009] Weissman, C. D. and Bobrowski, S. (2009). The design of the force.com multitenant internet application development platform. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pages 889–896, New York, NY, USA. ACM.

[Yang et al. 2009] Yang, F., Shanmugasundaram, J., and Yerneni, R. (2009). A scalable data platform for a large number of small applications. In *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, Online Proceedings*, pages 1–10.

[Youseff et al. 2008] Youseff, L., Butrico, M., and Da Silva, D. (2008). Toward a unified ontology of cloud computing. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10.

[Zhang et al. 2010] 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.

Capítulo

5

Minerando a Web por meio de Grafos - da teoria às aplicações

Ana Paula Appel, Estevam Rafael Hruschka Junior

Abstract

The volume of data represented as graphs, such as complex networks, has been increased over the past years making possible the creation of a new research area called graph mining. The main source of graph data is the World Wide Web, being link page structure one of the most common targets in this area. In graph mining, some tasks as statistical properties, community detection and link prediction can be highlighted. This course aims to presenting a global view of graph mining area; focusing on graph from Web like social networks. Also, we will introduce some basic concepts and the main techniques of graph mining main areas.

Resumo

O crescimento do volume de dados modelados como grafos, como as redes complexas, motivou e impulsionou a criação de novas áreas de pesquisa como a mineração de grafos. Atualmente, tais dados são provenientes principalmente da Web e uma das principais fontes é a estrutura de links entre as páginas. Nesta nova área de pesquisa, algumas tarefas se destacam, a saber: a extração de propriedades estatísticas, a detecção de grupos (comunidades), a predição de ligações (arestas), entre outras. Este minicurso tem como objetivo apresentar uma visão geral da área de mineração de grafos, focando principalmente em grafos vindo da Web, como ocorre com as redes sociais, por exemplo. Além de introduzir os conceitos básicos da mineração de grafos, serão também apresentadas as principais áreas e técnicas de mineração de grafos.

5.1. Introdução

Atualmente a Web é uma das fontes de dados heterogêneas mais ricas e em crescente evolução. A Web abrange diversos tipos de dados como multimídia, texto (dados não estruturados), dados semi-estruturados, entre outros. Por essas características, a Web apresenta desafios e oportunidades para a mineração de dados e descoberta de informação

e conhecimento. Nos últimos anos a mineração de ligações em conjuntos de dados estruturados tem recebido uma grande atenção de pesquisadores da área, principalmente com relação às redes complexas.

Uma rede complexa é, normalmente, modelada como um grafo, ou seja, a rede complexa é representada através de um objeto matemático cujos nós, também chamados vértices, modelam elementos (que podem ser páginas web, pessoas, computadores) e as arestas modelam relacionamentos entre os nós.

Uma característica muito importante, nesta abordagem de mineração de dados, é que os grafos que representam problemas reais tendem a ser irregulares e por isso eles são chamados de redes complexas. A modelagem da Web através da abordagem de redes complexas tem sido foco de muitas pesquisas atualmente. Com o intuito de guiar o leitor interessado em mineração de grafos aplicada neste tipo de redes, este documento apresenta os conceitos básicos necessários para a compreensão e a execução de tarefas de mineração de dados estruturados vindos da Web e representados como redes complexas. Assim, vários domínios de aplicação podem ser abordados, tais como as redes sociais (Facebook, Flickr, Orkut, etc), ontologias, redes acadêmicas (DBLP, ARXIV) e a própria WWW - World Wide Web.

Outro aspecto bastante relevante e que teve papel de destaque no aparecimento desta nova área de pesquisa dentro da mineração de dados foi a limitação dos algoritmos tradicionais quando aplicados em dados modelados como redes complexas. Os algoritmos de mineração de dados tradicionais, tais como regras de associação, detecção de agrupamentos, classificação, entre outros, usualmente encontram padrões em relações únicas que armazenam uma coleção de instâncias independentes. Um desafio emergente para a descoberta de conhecimento é o problema de minerar coleções de dados que estão inter-relacionados. Tais inter-relações podem ser naturalmente representadas através de grafos e armazenadas em diversas relações. A Figura 5.1 apresenta o grafo que representa a rede Web do Google[1].

Grafos são representações convenientes para um conjunto numeroso de dados inter-relacionados, como ocorre nas redes complexas, representadas pelas redes sociais, redes de publicações científicas, autores vs. participação em conferencias, e muitos outros. Nas últimas décadas, o volume de dados representados como grafo (as redes sociais LinkedIn, Facebook e Flickr, por exemplo), tem aumentado exponencialmente fazendo com que houvesse uma mudança no paradigma de como as redes são analisas [Newman, 2003]. Assim, ao invés das redes serem analisadas de uma maneira centralizada, com nós e arestas sendo estudadas individualmente, as redes passaram ser analisadas de uma maneira mais geral por meio de propriedades estatísticas de larga escala.

O aumento no tamanho das redes complexas se deve ao fato, principalmente, destas redes serem construídas a partir de dados vindos da Web. Este dinamismo e a velocidade de crescimento destas massas de dados fazem com que a utilização de algoritmos tradicionais de mineração de dados não traga resultados satisfatórios nestes domínios. Com isso, a mineração de grafos tem se tornado essencial para extrair conhecimento de

[1]Disponível em:http://commons.wikimedia.org/wiki/Image:WorldWideWebAroundGoogle.png (Acesso em 09/08/2010)

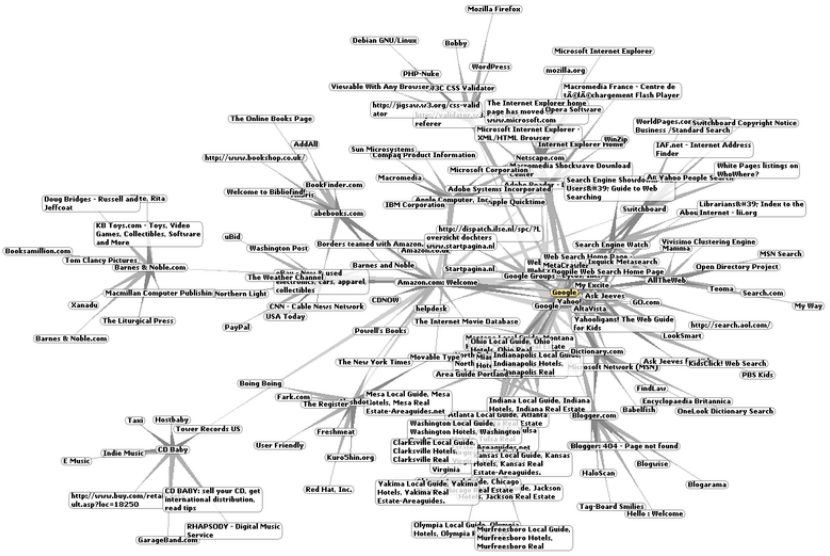


Figura 5.1. Rede Web da Google.

grandes redes complexas.

Muitas propriedades relevantes das redes complexas, algumas frequentemente não intuitivas, tais como diâmetro pequeno, distribuição do grau, triângulos, autovalores, presença de estrutura de comunidade, tem sido descobertas por meio da mineração das grandes redes complexas [Faloutsos et al., 1999, Leskovec et al., 2007, Clauset et al., 2007, Leskovec et al., 2008, Tsourakakis, 2008, McGlohon et al., 2008, Clauset et al., 2008]. Tais propriedades são importantes para o entendimento do comportamento e formação dessas redes. Além disso, essas propriedade provam que as redes complexas não são randômicas. Outro ponto é, que se a maioria dos nós de uma rede segue um padrão específico, o desvio de alguns nós desse padrão indica a presença de valores discrepantes ("outliers") que devem ser estudados.

5.2. Conceitos

As redes complexas, ou simplesmente redes, são um conjunto de elementos discretos que são representados pelos vértices e arestas, que são um conjunto de conexões entre os vértices. Os elementos e suas conexões podem representar, por exemplo, pessoas e ligações de amizade, computadores e linhas de comunicação [Faloutsos et al., 1999], componentes químicos e reações [Jeong et al., 2000], artigos e citações [Redner, 1998], entre outros. Assim, as redes complexas podem ser facilmente modeladas como um grafo.

Os grafos são capazes de abstrair os detalhes do problema ao descreverem características topológicas importantes com uma clareza que seria praticamente impossível se todos os detalhes fossem mantidos. Essa foi uma das razões por que a teoria dos grafos se espalhou, especialmente nos últimos anos, e tem sido utilizada por engenheiros, cientistas da computação e em especial por sociólogos.

Nesta seção serão apresentados os conceitos como os da teoria dos grafos, álgebra linear e outros que se fazem necessários para a o entendimento das tarefas de mineração

de grafos.

5.2.1. Teoria dos Grafos

Um grafo $\mathscr{G} = (\mathscr{V},\mathscr{E})$ é definido como um conjunto de nós $\mathscr{V}$ e um conjunto de arestas $\mathscr{E}$, sendo que $|\mathscr{V}| = N$ denota o número de nós e $|\mathscr{E}| = M$ denota o número de arestas, sendo $e_k \in \mathscr{E}$ e $e_k = \{(v_i, v_j)|v_i, v_j \in \mathscr{V}\}$. Os termos nó ou vértice são considerados sinônimos. Neste trabalho será usado o termo nó para referenciar os elementos do conjunto de vértices $\mathscr{V}$ e similarmente o termo aresta para referenciar os elementos do conjunto de arestas $\mathscr{E}$, que também é referenciado na literatura por meio dos seguintes sinônimos: *links*, *hops*, ligações ou conexões.

Uma maneira conveniente de representar um grafo $\mathscr{G}$ em um computador é usar uma matriz de adjacência, que é uma matriz $\mathbf{A}$ quadrada $N \times N$, sendo $N = |\mathscr{V}|$, em que $\mathbf{A}_{i,j} = 1$ se $(v_i, v_j) \in \mathscr{E}$ e 0 caso o contrário.

Tabela 5.1. Símbolos utilizados neste trabalho.

Símbolo	Descrição
$\mathbf{A}$	Matriz de Adjacência
$\mathscr{G}$	grafo
$\mathscr{G}_s$	subgrafo
$\mathscr{E}$	arestas
$\mathscr{E}_s$	arestas do subgrafo
$\mathscr{E}_{sp}$	arestas no `ShatterPoint`
$\mathscr{V}$	nós
$\mathscr{V}_s$	nós do subgrafo
$\mathscr{V}_{sp}$	nós no `ShatterPoint`
$\mathscr{D}$	diâmetro do grafo
$\mathscr{D}_e$	diâmetro efetivo do grafo
λ	autovalor do grafo
GCC	maior componente conexa do grafo
v_i	nó de um grafo
e_k	aresta de um grafo
Δ	triângulo
$d(v_i)$	grau do nó v_i
$d_{out}(v_i)$	grau de saída do nó v_i
$d_{in}(v_i)$	grau de entrada do nó v_i
d_{max}	maior grau do grafo
$P(v,u)$	caminho do nó v ao u
C(v_i)	coeficiente de clusterização do nó v_i
C($\mathscr{G}$)	coeficiente de clusterização do grafo
κ_t	clique de tamanho t
N	número de nós
M	número de arestas

A Tabela 5.1 apresenta os principais símbolos utilizados e a seguir apresentam-

se alguns conceitos básicos, extraídos de [Nicoletti, 2006, Bondy and Murty, 1979, Diestel, 2005], que serão usados neste minicurso:

- **Grafos Direcionados e Não Direcionados:** um grafo é *não direcionado* se $\{(v_i, v_j) \in \mathscr{E} \Leftrightarrow (v_j, v_i) \in \mathscr{E}\}$, isto é, as aresta são pares de nós sem ordem. Se um par de nós é ordenado, isto é, arestas tem direção, então o grafo é *direcionado*, também chamado de *dígrafo*.

- **Grau do Nó:** o nó v_i tem grau $d(v_i)$ se ele tem $|\mathscr{N}(v_i)|$ nós incidentes. Para grafos direcionados, o grau de um nó pode ser dividido em "grau de saída", $d_{out}(v_i)$ que é o número de arestas entram pelo nó v_i e "grau de entrada", $d_{in}(v_i)$ que é o número de arestas que saem para o nó v_i.

- **Triângulo:** em um grafo não direcionado um triângulo (Δ), também conhecido como fechamento transitivo, é uma tripla de nós conexos (u, v, w), tal que, $(u, v), (v, w), (w, u) \in \mathscr{E}$

- **Caminho:** é uma sequência de nós conectados entre si, $P(v_1, v_n) = (v_1, v_2, v_3, ..., v_n)$, tal que, entre cada par de nó existe uma aresta $(v_1, v_2), (v_2, v_3), ..., (v_{n-1}, v_n) \in \mathscr{E}$. Um caminho é **simples** se nenhum nó se repete. Dois caminhos são **independentes** se somente o primeiro e o último nó são comuns à eles.

- **Comprimento de um caminho:** é o número de arestas que o caminho contém. O **menor caminho** entre dois nós $P(v_i, v_j)$ é o caminho de menor número de arestas que ligam os dois nós.

- **Subgrafo:** um subgrafo $\mathscr{G}_s = (\mathscr{V}_s, \mathscr{E}_s)$ de um grafo $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ é um subconjunto de arestas e todos os nós tal que $\mathscr{E}_s \subseteq \mathscr{E} \Rightarrow \mathscr{V}_s = \{v_i, v_j | (v_i, v_j) \in \mathscr{E}_s\}$.

- **Grafo Conexo:** é um grafo que possui pelo menos um caminho entre todos os pares de nós.

- **Componente Conexa:** é o maior subgrafo, na qual existe um caminho entre qualquer par de aresta.

- **Grafo Induzido:** um subgrafo induzido $\mathscr{G}_s = (\mathscr{V}_s, \mathscr{E}_s)$ de um grafo $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ é um subconjunto de nós e todas as arestas que ligam este subconjunto de nós no grafo original $\mathscr{G}$, tal que $\mathscr{V}_s \subseteq \mathscr{V}$ e $\mathscr{E}_s = \{(v_i, v_j) | (v_i, v_j) \in \mathscr{E}_s \, v_i, v_j \in \mathscr{V}_s\}$.

- **Clique (κ_t):** é um subgrafo completo que possui um subconjunto de nós $\mathscr{V}_s \subseteq \mathscr{V}$ e arestas conectando todos os pares de nós em $\mathscr{V}_s$. O tamanho t do clique é definido pelo número de nós, $|\mathscr{V}_s| = t$. Um triângulo é um clique de tamanho 3 - κ_3.

- **Diâmetro:** o diâmetro $\mathscr{D}$ de um grafo $\mathscr{G}$ é o maior caminho dentre todos os menores caminhos existentes entre todos os pares de nós do grafo $\mathscr{G}$.

5.2.2. Leis de Potência

Uma distribuição que segue uma lei de potência é uma distribuição na forma:

$$p(x) = a * x^{-\gamma} \tag{1}$$

na qual $p(x)$ é a probabilidade de x ocorrer, sendo a uma constante de proporcionalidade e γ o expoente da lei de potência [Newman, 2005, Clauset et al., 2009].

Distribuições que seguem uma lei de potência ocorrem em muitas situações de interesse científico e são importantes para o entendimento de fenômenos naturais e humanos. A população das cidades e as intensidades dos terremotos são exemplos de fenômenos que têm a distribuição seguindo uma lei de potência.

5.2.3. Autovalores e autovetores

Em geral, uma matriz opera em um vetor transformando tanto a sua magnitude quanto a sua direção. Contudo, uma matriz pode operar em certos vetores transformando apenas a sua magnitude, deixando assim, a sua direção a mesma ou então transformando-a para o inverso. Estes vetores são chamados autovetores da matriz. A matriz opera em um autovetor pela multiplicação da sua magnitude por um fator, que se positivo sua direção não é alterada e se negativo sua direção é invertida. Este fator é chamado autovalor e está associado a um autovetor.

Essa transformação é chamada "transformação linear" e é formalmente definida como $A * x = \lambda * x$, sendo A a matriz de transformação linear, x o autovetor não nulo e λ o escalar. O escalar λ é considerado o autovalor de A correspondente ao autovetor x.

Autovalores e autovetores são conceitos importantes de matemática, com aplicações práticas em áreas diversificadas como mecânica quântica, processamento de imagens, análise de vibrações, mecânica dos sólidos, estatística, etc. Na seção 5.3.3 serão apresentados como os autovetores e autovalores podem ajudar na mineração de grafos.

5.3. Mineração de Grafos

Diversos domínios de aplicações têm seus dados modelados como redes complexas, por exemplo, a Internet, a World Wide Web (WWW), as redes sociais, de colaboração, biológicas, entre outras. Os pesquisadores nos últimos anos têm identificado classes de propriedades que podem ser encontradas em muitas das redes reais de vários domínios, sendo que muitas dessas distribuições seguem leis de potência, como a distribuição do grau dos nós, número de triângulos e os autovalores da matriz de adjacência da rede complexa.

5.3.1. Distribuição do grau do nó

Em redes complexas, é comum que a distribuição do grau dos nós siga uma lei de potência. Assim, a distribuição do grau dos nós de uma rede é uma lei de potência se o número de nós N_ϕ que possui um grau ϕ é dado por $N_\phi \propto \phi^{-y}$ $(y > 1)$, sendo $\mathscr{V}_\phi = \{v_i \in \mathscr{V} | d(v_i) = \phi\}$, $\left|\mathscr{V}_\phi\right| = N_\phi$ e y é chamado de expoente da distribuição do grau. A grande maioria das redes reais apresentam uma distribuição do grau dos nós que segue uma lei de potência, por isso são chamadas redes livres de escala.

Uma distribuição livre de escala, significa intuitivamente que uma distribuição se parece com ela mesma independente da escala em que se esta olhando. A noção de auto similaridade é implícita no nome "livre de escala". A auto similaridade de elementos consiste no fato do elemento manter as mesmas propriedades seja qual for a escala utilizada [Schroeder, 1991].

Este tipo de distribuição tem sido encontrada em grafos de ligações telefônicas [Abello et al., 1998], na Internet [Faloutsos et al., 1999], na Web [Kleinberg et al., 1999, Broder et al., 2000, Flaxman et al., 2005, Huberman and Adamic, 1999, Kumar et al., 1999], em grafos de citações [Redner, 1998], *click-stream* [Bi et al., 2001], em redes sociais online [Chakrabarti et al., 2004b] e muitas outras.

Tipicamente, para muitos conjuntos de dados, o expoente da distribuição do grau tem valor $2 < y < 3$. Por exemplo, a distribuição do grau de entrada da Web é $y_{in} = 2,1$ e de saída $y_{out} = 2,4$ [Reka and Barabási, 2002], enquanto para as redes de computadores chamadas de Sistemas Autônomos $y = 2,4$ [Faloutsos et al., 1999]. Contudo, alguns desvios do padrão da lei de potência foram notados em [Pennock et al., 2002].

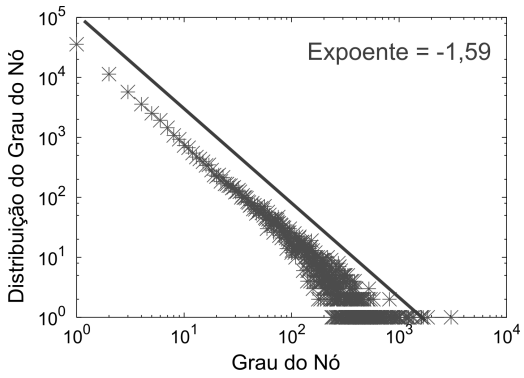


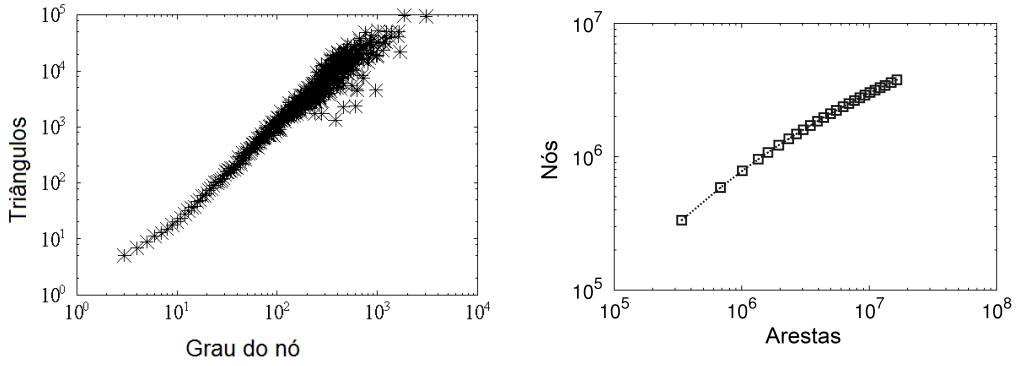
Figura 5.2. Gráfico da distribuição do grau dos nós da rede Epinions (quem confia em quem).

Além da distribuição do grau dos nós as seguintes distribuições tendem a seguir as leis de potência: número de triângulos em relação ao grau dos nós [Tsourakakis, 2008] e crescimento do número de nós e arestas na evolução das redes [Leskovec et al., 2007] entre outras. A Figura 5.3 apresenta os gráficos de algumas dessas distribuições. Um fato interessante sobre a distribuição dos triângulos em relação ao grau do nó é que a sua distribuição tem expoente oposto a distribuição do grau do nó.

5.3.2. Diâmetro efetivo

O diâmetro $\mathscr{D}$, como definido anteriormente, é o maior caminho dentre todos os menores caminhos existentes entre todos os pares de nós do grafo $\mathscr{G}$. O diâmetro também é referenciado como *diâmetro completo*. Para grafos com mais de uma componente conexa, o diâmetro usualmente é definido como infinito. Além disso, o diâmetro é suscetível aos efeitos degenerativos da estrutura do grafo como por exemplo, o surgimento de um caminho muito longo no grafo durante a sua evolução.

Calcular o diâmetro de um grafo grande é computacionalmente



(a) Δ vs. Grau do Nó (b) # Nós vs. # Arestas

Figura 5.3. Gráficos de outras distribuições que seguem lei de potência. Em (a) a distribuição dos triângulos versus o grau do nó da rede Epinions [Tsourakakis, 2008]. Em (b) a quantidade de nós versus a quantidade de arestas da rede Patente-US durante o seu crescimento [Leskovec et al., 2007].

caro(complexidade de tempo $O(N^3)$). Uma maneira mais eficiente de realizar este cálculo é pela amostragem de nós, isto é, uma quantidade de nós é amostrada e então o diâmetro é calculado entre os pares de nós amostrados [Albert et al., 1999]. Uma outra abordagem é usar o algoritmo de aproximação ANF [Palmer et al., 2002] que é baseado no cálculo aproximado chamado diâmetro efetivo. Define-se o diâmetro efetivo como sendo o menor número de "arestas" em que no mínimo 90% de todos os nós da maior componente conexa do grafo podem ser alcançados entre si.

O diâmetro efetivo é um valor mais robusto que o diâmetro, pois, nele somente os pares de nós conexos são considerados e a direção das arestas (no caso de grafos direcionados) são ignoradas. Além disso, muitos experimentos mostram que o diâmetro efetivo e o diâmetro exibem comportamento qualitativamente similar.

Em detalhes, seja $g_h(\mathscr{G})$ uma função que calcula para cada nó v_i o número de nós que tenham um caminho de máximo h arestas de distância partindo de cada nó v_i, sendo $i = 1, ..., N$. O valor de h inicia-se em h=1 até que $g_{(h-1)}(\mathscr{G})$-$g_h(\mathscr{G})$<$threshold$. Para h=1 tem-se $g_h(\mathscr{G})$= $\sum_{i=1}^{N} |d(v_i)|$ =$|\mathscr{E}|$. O gráfico que representa a função $g_h(\mathscr{G})$ é chamado "Hop-Plot" e é apresentado na Figura 5.4 [Leskovec et al., 2005]. A função $g_h(\mathscr{G})$ foi aplicada a uma rede complexa. O diâmetro efetivo desta rede é igual a $\mathscr{D}_e = 5$ sendo representado pela linha tracejada no gráfico apresentado na Figura 5.4.

Definindo mais formalmente o diâmetro efetivo tem-se: o diâmetro efetivo de $\mathscr{G}$ é definido como sendo $\mathscr{D}_e = h$ sendo que $g_h(\mathscr{G})$ representa 90% do número de pares de nós alcançáveis.

Muitos dos grafos reais exibem um diâmetro relativamente pequeno, conhecido como o fenômeno "Small-World"[Milgram, 1967]. Por exemplo, o diâmetro efetivo é pequeno para grandes redes reais, tais como a Internet, a Web, as redes sociais [Reka and Barabási, 2002, Barabasi and Albert, 1999, Broder et al., 2000, Bollobás and Riordan, 2004, Chung et al., 2002].

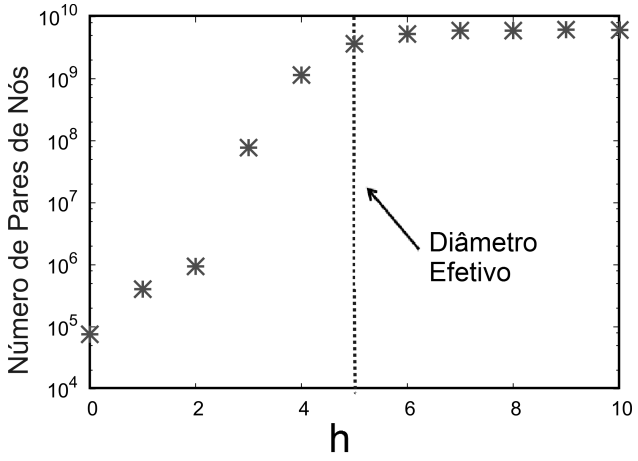


Figura 5.4. Gráfico Hop-Plot de uma rede complexa. A linha tracejada em 5 representa o Diâmetro efetivo da rede.

5.3.3. Autovalores e autovetores para grafos

Na teoria dos grafos, os autovalores λ_i de um grafo $\mathscr{G}$ são definidos como sendo os autovalores da matriz de adjacência A do grafo $\mathscr{G}$ ou a matriz Laplaciana B. Para A tem-se que $\mathbf{A}_{i,j} = 1$ se $(v_i, v_j) \in \mathscr{E}$ e 0 caso o contrário. Para B tem-se que $\mathbf{A}_{i,j} = d(v_i)$ se $i = j$, $\mathbf{A}_{i,j} = -1$ se $i \neq j \wedge (v_i, v_j) \in \mathscr{E}$ e 0 caso o contrário. Assim, como a matriz de adjacência, a matriz laplaciana pode ser usada para encontrar muitas propriedades dos grafos. As propriedades de ambas as matrizes fazem parta da teoria do *spectrum* do grafo. A seguir serão apresentadas algumas propriedades. Note-se que quando usado A será considerada a matriz de adjacência e quando usado B a matriz laplaciana.

Para todas as definições e teoremas que serão apresentados a seguir tem-se que o grafo $\mathscr{G}$ é não direcionado, com isso a matriz é simétrica. As provas dos teoremas e definições podem ser encontradas em [Mihail and Papadimitriou, 2002, Chung, 1994].

Seja A uma matriz de adjacência quadrada $N \times N$ com os seguintes autovalores λ_i, sendo $i = 1, 2, 3, ..., N$. Então:

Teorema 1 *Para um grafo $\mathscr{G}$ com $|\mathscr{V}| = N$ nós, tem-se que $\lambda_1(\mathscr{G}) \geq \lambda_2(\mathscr{G}) \geq \lambda_3(\mathscr{G}) \geq ... \ \lambda_N(\mathscr{G})$ são os autovalores da matriz de adjacência do grafo $\mathscr{G}$ em ordem decrescente.*

Assim, $\lambda_1(\mathscr{G})$ é o maior autovalor de $\mathscr{G}$, também chamado de principal autovalor.

Teorema 2 *Para um grafo $\mathscr{G}$ com N nós e $|\mathscr{E}| = N - 1$ tem se que $\lambda_1 \cong \sqrt{d_{max}}$.*

Teorema 3 *Um grafo $\mathscr{G}$ conexo é bipartido se $-\lambda_1$ também é um dos seus autovalores.*

Teorema 4 *Um grafo $\mathscr{G}$ a $\sum_{i=1}^{N} {\lambda_i}^2 = |\mathscr{E}|$.*

Seja B uma matriz de laplaciana quadrada $N \times N$ com os seguintes autovalores λ_i, sendo $i = 1, 2, 3, ..., N$. Então:

Teorema 5 *Se $\mathscr{G}$ é conexo então $\lambda_1 > 0$. Se $\lambda_i = 0$ e $\lambda_{i+1} \neq 0$, então $\mathscr{G}$ tem exatamente $i+1$ componentes conexas.*

O segundo autovetor da matriz laplaciana é largamente utilizado como método de bisseção de grafos para encontrar comunidades, como será apresentado na seção 5.3.5. Os autovetores e autovalores da matriz de adjacência de um grafo podem ser utilizados para diversas tarefas, como por exemplo na área de epidemiologia para indicar se a ocorrência de uma doença (modelada por uma rede complexa) está próxima ou não de uma epidemia, para isso basta comparar o número de morte do vírus versus o número de nascimento do vírus com $1/\lambda_1$ que é chamado de *Epidemic Threshold* [Chakrabarti et al., 2008]. Além disso, como será apresentado na seção 5.3.4 os autovalores também podem ser usados para calcular o número de triângulos aproximado de um grafo.

Um outro exemplo de aplicação de autovalores e autovetores é o algoritmo PageRank do Google [Page et al., 1998], que utiliza o primeiro autovetor da matriz de adjacência modificada de um grafo para fazer o *rank* de cada um dos nós. Considerando que o principal objetivo do PageRank é a ordenação de páginas Web para consultas, os nós do grafo são páginas e as aresta os *links* entre as páginas. Neste algoritmo também há um fator chamado *damping factor* - *df* - que é a probabilidade de um usuário encerrar a consulta e este valor é geralmente $0,85$. A Equação 2 representa o cálculo original do PageRank para todos os N nós de uma rede.

$$PR_{t+1} = \frac{1-df}{N} + df * A * PR_t \tag{2}$$

Neste algoritmo, a matriz de adjacência A é primeiro normalizada, isto é, cada elemento de uma linha da matriz é dividido pelo quantidade de "uns" existente na linha, sendo o somatório de cada linha igual a um. Na iteração inicial tem-se: $A(i,j) = \frac{1}{d_{out}(v_i)}$ se $\exists(v_i, v_j)$ senão $A(i,j) = 0$. O vetor PR é inicializado todo com 1.

5.3.4. Triângulos e coeficiente de clusterização

Em muitas redes, especialmente as redes sociais, é notado que se um nó u é conectado com um nó v que é conectado com w, então há uma grande probabilidade de u ser conectado com w. Esta relação é chamada de transitividade e é medida pelo coeficiente de clusterização [Watts and Strogatz, 1998]. A transitividade significa a presença de um alto número de triângulos ($\Delta(v_i)$) na rede. A contagem de triângulos é a principal parte do coeficiente de clusterização, que pode ser calculado para cada nó do grafo (Equação 3) ou para o grafo como um todo (Equação 4). Este coeficiente tem o objetivo de indicar quão próximo o grafo está de ser um grafo completo. O coeficiente de clusterização $C(v_i)$ de um nó v_i de grau $d(v_i)$ é definido pela Equação 3 a seguir.

$$C(v_i) = \frac{2 * \Delta(v_i)}{d(v_i) * (d(v_i) - 1)} \tag{3}$$

Seja v_i um nó com grau $|d(v_i)|$, então no máximo $d(v_i) * (d(v_i) - 1)/2$ arestas podem existir entre eles, sendo $\Delta(v_i)$ a fração de arestas que realmente existe, isto é o

número de triângulos. Isto significa que, o coeficiente de clusterização $C(v_i)$ de um nó v_i é a proporção de arestas entre os nós da sua adjacência dividido pelo número de arestas que podem existir entre eles. Equivalentemente, $C(v_i)$ é a fração de triângulos centrados no nó v_i entre $(d(v_i) * (d(v_i) - 1))/2$ triângulos que possam existir.

O coeficiente de clusterização global $C(\mathscr{G})$ é a média da soma de todos os $C(v_i)$ dos nós do grafo $\mathscr{G}$, dividido pelo número total de nós N. A equação do coeficiente de clusterização global é apresentada na Equação 4 a seguir.

$$C(\mathscr{G}) = \frac{1}{N} * \sum_{i=1}^{N} C(v_i) \tag{4}$$

Encontrar a quantidade de triângulos que cada nó possui, bem como a quantidade total de triângulos no grafo é um processo computacionalmente caro. A sua complexidade é de $O(N^2)$, sendo N o número total de nós do grafo.

Para reduzir essa complexidade, em alguns trabalhos como em [Latapy, 2008] os autores propõem algumas otimizações para a contagem e listagem dos triângulos em um grafo. Alguns trabalhos contam triângulos sem identificá-los [Bar-Yossef et al., 2002, Becchetti et al., 2008, Tsourakakis, 2008]. Sendo [Tsourakakis, 2008] o trabalho que apresenta uma melhor aproximação quanto ao número de triângulos e complexidade computacional. Neste trabalho é comprovado que o número total de triângulos é proporcional a soma dos autovalores da matriz de adjacência do grafo elevado ao cubo. A Equação 5 que representa esta proposição é apresentada a seguir:

$$\Delta(\mathscr{G}) = \frac{1}{6} * \sum_{i=1}^{N} \lambda^3{}_i \tag{5}$$

5.3.5. Detecção de Comunidades

Na sociedade há uma grande variedade de possíveis organizações em grupos: família, trabalho e circulo de amizade, cidades, nacionalidades. A difusão da internet tem direcionado a criação de grupos virtuais, que vivem na Web como as comunidades online Facebook, Orkut, LinkedIn. A detecção de comunidades pode ter diversas aplicações reais. Agrupar clientes web que tem interesses similares e são geograficamente próximos entre si pode aumentar a performance dos serviços oferecidos na WWW, em que cada grupo de cliente pode usar um servidor dedicado [Krishnamurthy and Wang, 2000]. Identificar grupos de clientes com interesses similares no relacionamento de compras entre produtos e consumidores de lojas onlines, como Amazon, permite o desenvolvimento de um sistema de recomendação eficiente, que ajuda a guiar consumidores nas lista de itens das lojas e permite uma oportunidade de negócio [Reddy et al., 2002].

Grupos de nós que tendem a ser mais conectados entre si que com o restante da rede são chamados de grupos ou comunidades. Pessoas tendem a formar comunidades, isto é, grupos pequenos no qual todo mundo conhece praticamente todo mundo. Com isto, grupos de nós pertencentes a uma mesma comunidade tendem a ter um número elevado de triângulos. Além disso, os membros das comunidades têm pouco relacionamento com

membros fora das comunidades que participam e cada um dos grupos tendem a estar organizados hierarquicamente, isto é comunidades dentro de comunidades.

Uma grande quantidade de algoritmos têm sido desenvolvidos para definir e identificar comunidades em redes sociais e de informações [Girvan and Newman, 2002, Radicchi et al., 2004]. Muitas vezes também é assumido que as comunidades obedecem a uma estrutura recursiva, em que grandes comunidades podem futuramente ser divididas em comunidades menores [Guimerà et al., 2007, Clauset et al., 2004].

A ideia do algoritmo de identificação de comunidades é particionar a rede em subgrafos menores por meio da remoção de um número mínimo de arestas. As comunidades encontradas podem ter seus elementos disjuntos, isto é, um nó que participa em um grupo não deve participar em outro, aceitando-se apenas algumas sobreposições [Fortunato, 2010]. Contudo, o contrário também pode ser encontrado, isto é, um mesmo nó faz parte de mais de uma comunidade. Esta situação pode ser vista na vida real, por exemplo: as pessoas nem sempre participam de apenas um grupo social. Boas revisões sobre detecção de comunidades podem ser encontradas em [Fortunato, 2010, Leskovec et al., 2008].

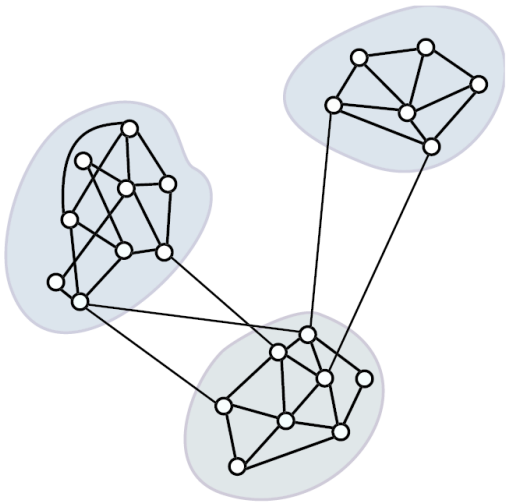


Figura 5.5. Grafo apresentando 3 comunidades do modo tradicional.

A estrutura de comunidades, como exemplificado pela Figura 5.5, foi observada também em diversas redes reais, como [Newman, 2006, Ravasz et al., 2002, Girvan and Newman, 2002]. Entretanto, esta forma de organização das redes complexas parece não valer para as redes complexas volumosas, com centenas de milhares de nós, como mostrado em [Leskovec et al., 2008], que demonstra que esta estrutura hierárquica está presente somente nas redes complexas pequenas, isto é redes com poucas centenas de nós.

As grandes redes complexas tendem a não possuir conjuntos de nós interligados formando comunidades muito bem definidas (Figura 5.5). Na verdade, as grandes redes tendem a apresentar uma estrutura chamada "Centro-Periferia" [Borgatti and Everett, 1999, Holme, 2005], que em ciência da computação é conhecida também pelo nome "*jellyfish*" [Tauro et al., 2001] ou "*octopus*" [Chung and Lu, 2006] e

que é exemplificada na Figura 5.6. Este conceito significa que uma rede é composta por um grande e denso conjunto de nós (core/centro) ligados entre si que basicamente não tem nenhuma estrutura de comunidade hierárquica, isto é, não podem ser quebrados em comunidades menores. Assim, a estrutura Centro-Periferia sugere o oposto da estrutura de comunidade hierárquica, e parece ser o mais encontrado em redes complexas de grande escala [Leskovec et al., 2008] e também em redes de computadores chamados Sistemas Autônomos [Siganos, 2006].

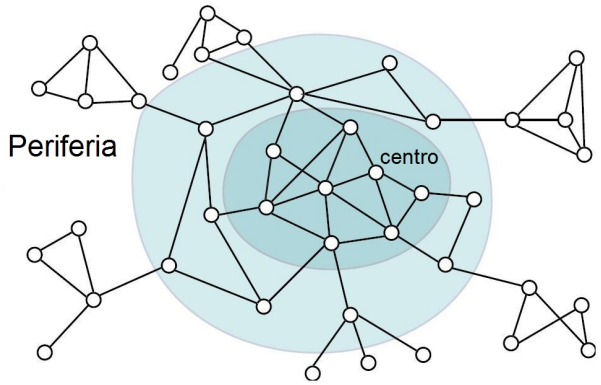


Figura 5.6. Grafo apresentando a topologia "Centro-Periferia"[Leskovec et al., 2008].

Também em [Leskovec et al., 2008] os autores mostram que as comunidades tendem a ser pequenas, com não mais do que 100 nós, e pouco conectadas com o restante da rede. O valor 100 é conhecido como o número de Dunbar, que é o número máximo de relacionamentos que uma pessoa consegue administrar [Dunbar, 1998].

A detecção de grupos (comunidades) não é importante apenas para redes sociais mas para inúmeras áreas da computação. Um exemplo, na computação paralela, é a determinação da melhor maneira de distribuir as tarefas para minimizar a comunicação entre os processadores, sendo as técnicas principais as baseadas no particionamento da rede. O problema de particionar um grafo consiste em dividir o conjunto de nós em k grupos de tamanho pré-definido, tal que, o número de arestas entre cada um dos grupos é mínimo. Cada grupo é chamado *cluster* ou comunidade e o número de arestas removidas é chamado *cut size*. Especificar o número de grupos em que o grafo será particionado é necessário, pois se este número fosse deixado livre a resposta trivial seria uma única partição com todos os nós. A Figure 5.7 ilustra o particionamento de um grafo em dois grupos ($k = 2$) cada um com 7 nós.

Um algoritmo muito tradicional para o particionamento de redes é o METIS, que permite o particionamento do grafo em k grupos. O algoritmo METIS funciona da seguinte maneira: dado um grafo $\mathscr{G}$, este é reduzido para um grafo com o agrupamento dos nós adjacentes em um mesmo nó. A bisseção deste grafo muito menor é computada e então o particionamento é projetado no grafo original por meio de refinamentos periódicos da partição [Karypis and Kumar, 1998]. A Figura 5.8 ilustra a aplicação do algoritmo METIS em um grafo. Note que apesar de na ilustração ser mostrado apenas o particionamento em dois, o METIS permite que o grafo seja particionado em k grupos, sendo k definido pelo usuário.

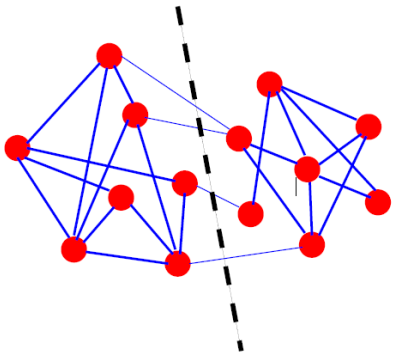


Figura 5.7. Um grafo sendo particionado em dois grupos iguais, cada um com 7 nós.

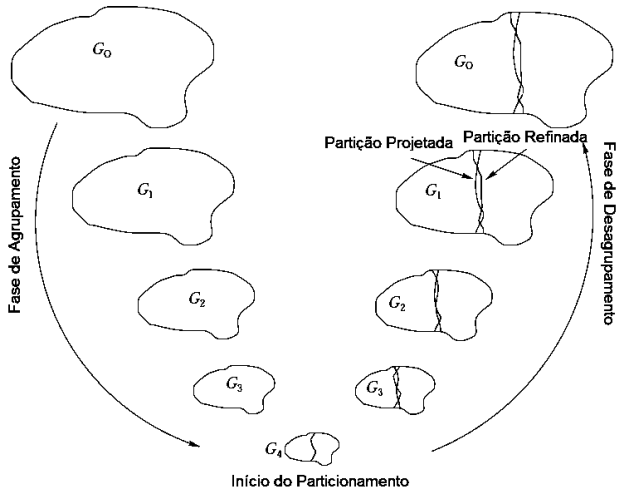


Figura 5.8. Exemplo de funcionamento do algoritmo METIS [Karypis and Kumar, 1998].

Outro método utilizados para o particionamento de grafos é o *Spectral Bisection* [Alon, 1998]. Este método é baseado no cálculo do segundo menor autovetor da matriz laplaciana da rede. Com este autovetor a matriz é reordenada e os grupos são identificados. Um exemplo deste método é apresentado na Figura 5.9. Primeiro a matriz de adjacência de um grafo $\mathscr{G}$ ao seu lado a matriz é reordenada pelo segundo menor autovetor da matriz laplaciana do grafo.

Um dos primeiros algoritmos realmente voltados para a identificação de comunidades em redes sociais foi o de Girvan e Newman [Girvan and Newman, 2002]. Neste algoritmo para cada aresta é calculado uma medida, chamada *betweenness*, que contabiliza o poder de "quebrar" de cada aresta. Esta medida calcula para cada aresta a quantidade de caminhos mínimos que utilizam esta aresta para interligar dois nós. Assim, quanto maior a quantidade de caminhos mínimos entre dois pares nós que contenham esta aresta, maior será o seu *betweenness*. Após o cálculo do *betweenness* para cada aresta, a aresta como o maior *betweenness* é removida da rede e então as arestas afetadas por esta remoção tem o seu *betweenness* recalculado. Este procedimento se repete até não haver mais arestas no grafo. O *betweenness* também pode ser calculado em relação a um nó da rede. O *betweenness* é uma medida muito cara computacionalmente já que é baseada no

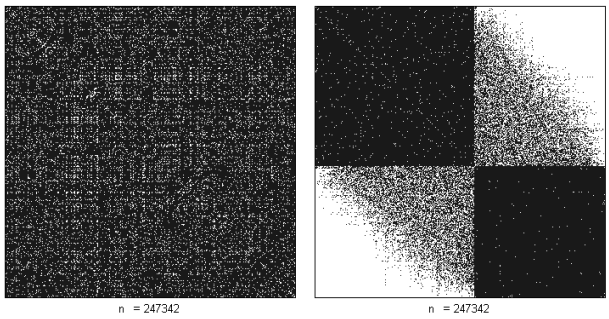


Figura 5.9. Matriz de adjacência de um grafo $\mathscr{G}$ e a matriz reordenada pelo segundo menor autovetor da matriz laplaciana do grafo $\mathscr{G}$

cálculo de caminhos mínimos o que a torna pouco aplicável a grande redes.

Os métodos supracitados não permitem a sobreposição de comunidades. A sobreposição é uma característica importante, principalmente nas redes sociais, já que, as pessoas naturalmente participam de mais de um grupo, como, escola, esportes, etc. Assim, um método bastante interessante que permite a sobreposição é o método Cross-Association [Chakrabarti et al., 2004a]. Este método faz uma decomposição conjunta da matriz de adjacência em grupos de linhas e colunas disjuntas, tal que intersecções retangulares são grupos homogêneos. O método é baseado em permutação de linhas e colunas utilizando-se do princípio MDL (Minimum Description Language). A ideia principal é que a matriz binária de uma rede representa a associação entre objetos (linhas e colunas) e quer se encontrar associações cruzadas entre esses objetos, isto é grupos homogêneos retangulares.

A Figura 5.10 apresenta a matriz de adjacência de duas redes reais Epinions e Oregon, processadas por este método. Quanto mais escura a área retangular mais denso é o grupo encontrado. Uma vantagem desse método é que o número de grupos é encontrado pelo método, além disso o número de grupos não precisa ser o mesmo para linhas (K) e colunas (l).

5.3.6. Resistência a ataques

Uma questão que tem sido considerada como foco de pesquisa recente é a definição de quão robusta é uma rede. Como visto anteriormente, uma rede pode representar o relacionamento interpessoal, as pessoas seriam os nós e o relacionamento as arestas. Imagine por exemplo que os nós da rede sejam marcados ou removidos conforme as pessoas contraiam uma doença ou tenham acesso a uma informação. Este tipo de análise é conhecido como resistência a ataques.

A Internet, por exemplo, é altamente robusta, já que há diferentes rotas ligando os computadores e/ou roteadores (nós), fazem com que a informação possua caminhos independentes para navegar de um nó para o outro. Assim, mesmo se um roteador falhar, o sistema é capaz de refazer a rota e fazer com que a informação chegue ao seu destino. De fato, na Internet sempre haverá um conjunto de roteadores que não estará funcionando em um dado momento. Entretanto, o importante é que a Internet como um todo continuará funcionando mesmo com uma grande quantidade de falhas [Wu et al., 2007].

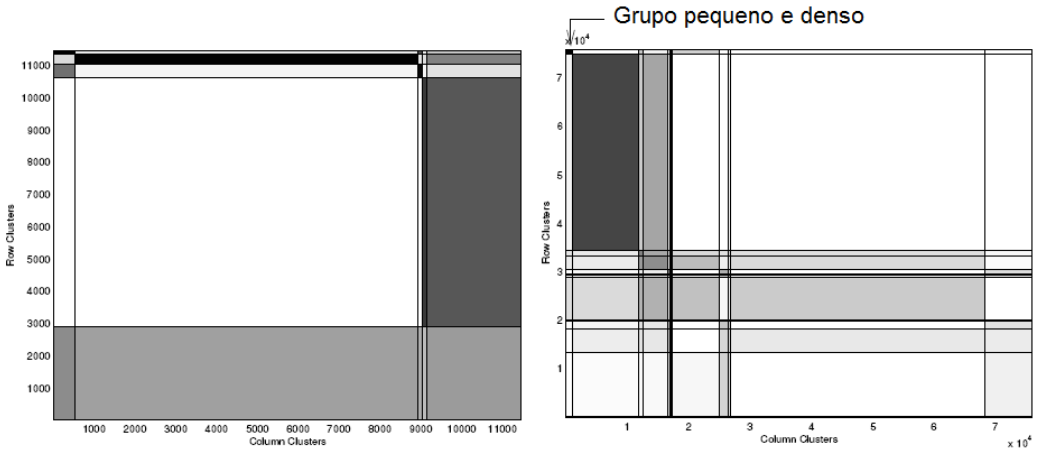
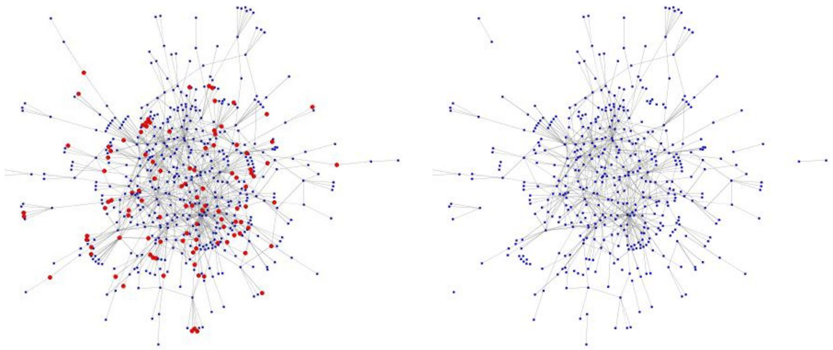


Figura 5.10. Duas redes reais, Epinions e Oregon, processadas pelo método Cross-Association [Chakrabarti et al., 2004a].



(a) (b)

Figura 5.11. Exemplo de uma rede complexa que tem 20% dos seus nós removidos aleatóriamente: (a) é a rede original e os nós marcados em vermelhos são os nós que serão removidos e (b) é a rede (a) já com os nós removidos.

Descobrir qual é o efeito da falha de uma certa fração de nós na conectividade do restante da rede tem sido o objetivo de inúmeros pesquisadores. Se ao invés da rede representar computadores ela representar pessoas, as falhas em uma rede podem representar a quantidade de pessoas que contraíram alguma doença, por exemplo, uma gripe.

Se os nós de uma rede são removidos de modo aleatório, como mostrado na Figura 5.11, o efeito é usualmente pequeno. Isto se deve ao fato de que as redes possuem um alto número de nós de grau ”um” e, em uma remoção aleatória, esses nós teriam uma alta probabilidade de serem removidos. A remoção de nós de grau um não altera a estrutura da rede já que eles se encontram na periferia da rede. A Figura 5.11 (b) apresenta a remoção aleatória de 20% dos nós de uma rede e como é notado, a rede continua robusta. Contudo, se os nós são removidos de um modo cuidadoso, isto é, segundo alguma ordenação que não seja aleatória, o dano causado na rede pode ser grande. Em [Albert et al., 2000, Cohen et al., 2000] a resistência das redes é analisada segundo a remoção aleatória, chamada de **falha** e a remoção baseada na ordenação dos nós pelo seu

grau, chamada de **ataque**.

As redes livres de escala são muito resistentes a falhas (remoção aleatória de nós), mas elas são substancialmente menos robusta quanto a um ataque. Isto acontece por que durante as falhas, uma grande quantidade de nós removidos são nós de grau um, já que estes são maioria nas redes reais. Já os ataques que removem os nós de alto grau, fazem com que a rede se torne desconexa muito rapidamente. Um exemplo do comportamento de uma rede real é ilustrado na Figura 5.12. Neste exemplo o diâmetro é avaliado quanto a quantidade de aresta após a deleção de nós. Note-se que, na remoção dos nós de alto grau o diâmetro cresce rapidamente e depois decai, mostrando que a rede torna-se desconexa muito rapidamente. Já na deleção de nós de baixo grau o diâmetro apresenta uma certa constância no seu valor.

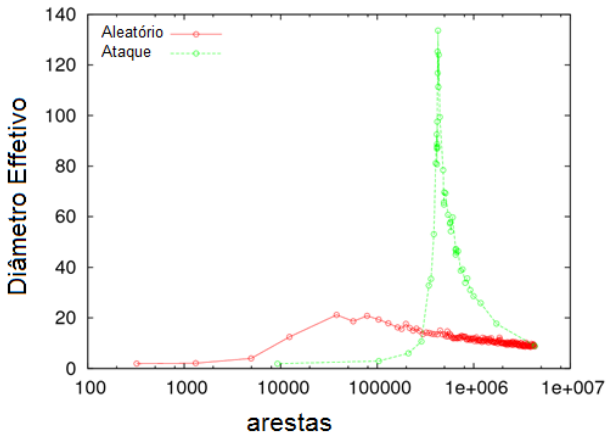


Figura 5.12. Gráfico mostrando o comportamento de uma rede real quanto a remoção de nós, ataque e falha.

As falhas e ataques às redes são geralmente analisadas quanto ao diâmetro e ao tamanho (número de nós) da maior componente conexa. O diâmetro aumenta conforme mais nós são removidos, mas esse aumento é mais lento nas redes livres de escala do que nas randômicas. Similarmente, o tamanho da maior componente conexa (GCC) decai mais devagar nas redes livres de escala do que nas randômicas. Este comportamento é ilustrado na Figura 5.13. As circunferências representam o tamanho da rede quanto as suas componentes conexas, as circunferências menores representam as componentes conexas menores.

Como ilustrado, as redes randômicas apresentam o mesmo comportamento quanto a falhas e ataques. Isso acontece pois as redes randômicas não tem a distribuição de grau seguindo uma lei de potência, na verdade, os nós têm o grau muito próximo a média do grau dos nós da rede.

O algoritmo `ShatterPlots` [Appel et al., 2009] encontra padrões em redes complexas, reais e sintéticas, por meio da remoção de arestas. Intuitivamente, dada uma rede $\mathscr{G}$ a cada passo t, Q_t arestas são escolhidas aleatoriamente para serem removidas da rede. Após cada remoção um conjunto de medidas, como diâmetro efetivo, autovalor da matriz de adjacência, número de nós e arestas são coletados da rede e o processo continua até todos os nós estarem isolados.

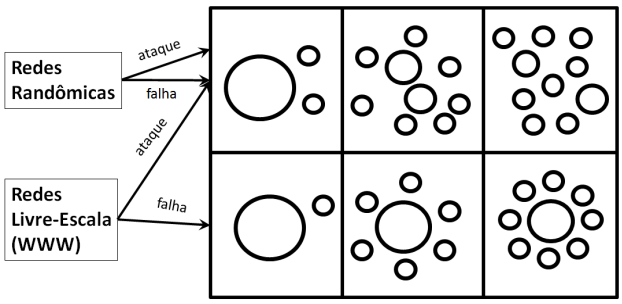


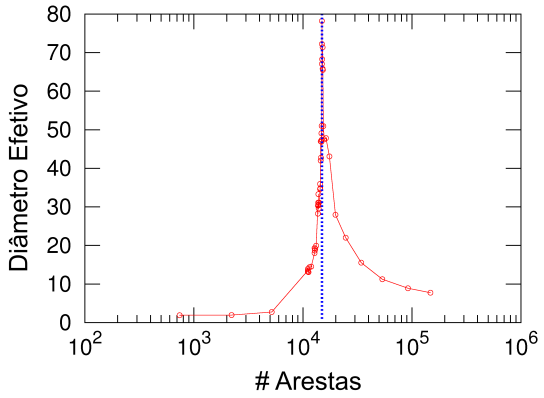
Figura 5.13. Comparação do comportamento do tamanho da rede quanto a remoção de nós aleatória e nós de alto grau. [Albert et al., 2000]

A Figura 5.14 apresenta medidas diâmetro efetivo, número de pares alcançáveis e número de nós pertencentes à maior componente conexa (GCC) na rede real Gnutella [Ripeanu et al., 2002] durante a remoção aleatória de arestas pelo algoritmo `ShatterPlots`. Todas as medidas são em relação a quantidade de arestas existente na rede. Como pode ser observado, apenas o diâmetro mostra um pico, chamado `ShatterPoint` e marcado por uma linha, nos gráficos apresentados na Figura 5.14. As outras medidas também possuem uma fase de transição, que é a mesma do diâmetro e que também esta marcada com uma linha azul, entretanto, elas não podem ser identificadas de maneira clara nos gráficos.

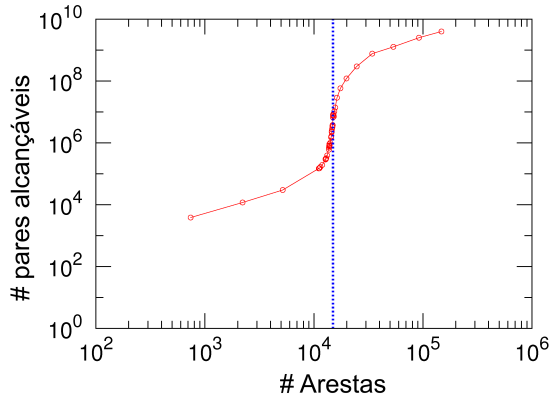
O algoritmo `ShatterPlots` é um método adaptativo, pois ele ajusta o número de arestas removidas na rede. Este ajuste faz com que o número de arestas removidas diminua se o diâmetro aumentar mais que um certo *threshold* e aumente se o diâmetro se manteve estável. Isto torna constante o número de iterações do algoritmo, fazendo com que ele seja escalável quanto ao número de arestas.

Dentre os padrões encontrados pelo `ShatterPlots` no `ShatterPoint` dois se destacam. O primeiro é o chamado *30-per-cent*. Este padrão revela que para todas as redes, sintética e reais, o `ShatterPoint` ocorre sempre na mesma proporção de nós e aresta, isto é, ele ocorre quanto o número de nós não isolados da rede é 30% maior que o número de arestas atual da rede ($\mathscr{V}_{sp} = 1.30 * \mathscr{E}_{sp}$). Este valor é conhecido na teoria dos grafos randômicos, *Erdös-Rényi*, como fase de transição ou *percolation*. Esta fase é na teoria conhecida como o ponto em que uma rede passa de desconexa para conexa, isto é, abaixo do ponto de transição a rede possui pequenas componentes conexas e acima deste ponto a rede possui uma grande componente conexa. Quanto mais acima deste ponto uma rede estiver mais bem conexa ela será. A Figura 5.15 apresenta o gráfico que mostra o padrão *30-per-cent*. Neste gráfico, os triângulos representam as redes sintéticas, entre elas *Preferential Attachment*, *Small-World* e *Erdös-Rényi*. Os demais símbolos representam as redes reais, dentre elas, Gnutell, Amazon, Oregon, etc.

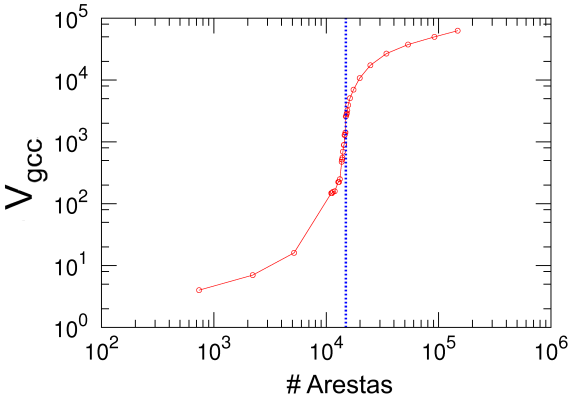
O segundo é o padrão chamado *NodeShatteringRatio* . Este padrão revela que em um gráfico com o número atual de nós da rede ($|V_{sp}|$) versus o número original de nós da rede ($|V|$) é possível traçar uma linha ($\mathscr{V}_{sp} = 0.37 * \mathscr{V}$) em que, acima desta linha todas as redes são sintéticas e abaixo dela todas são reais. A Figura 5.16 apresenta o gráfico do padrão *NodeShatteringRatio* . Mais detalhes podem ser encontrados em [Appel et al., 2009].



(a) Diâmetro Efetivo



(b) # pares alcançáveis



(c) # nós da GCC

Figura 5.14. Medidas estruturais da rede feitas durante a remoção aleatória de arestas. A rede apresenta `ShatterPoint` em todas as medidas mas somente o diâmetro apresenta um pico.

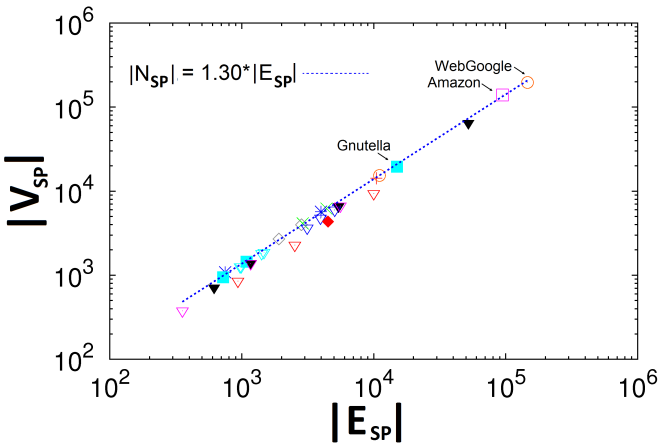


Figura 5.15. O padrão *30-per-cent* para redes reais e sintéticas (triângulos).

5.3.7. Predição de Ligações

A predição de ligações pode ser definida como, dado um "*snapshot*" de uma rede complexa em um tempo t, quer se prever com uma certa acurácia as arestas que irão surgir na rede complexa no tempo futuro $t+1$.

Dentre as técnicas de predição de ligação destacam-se as baseadas em propriedades estruturais do grafo [Liben-Nowell and Kleinberg, 2003, Huang, 2006]. Um exemplo interessante é apresentado em [Clauset et al., 2008], em que a descoberta de grupos, isto é comunidades, em redes complexas é usado para o auxílio na identificação de ligações faltantes, já que pares de nós pertencentes a uma mesma comunidade têm mais chance de serem conexos entre si do que pares de nós pertencentes a comunidades diferentes. Este método se diferencia da predição de ligação tradicional, pois normalmente esta tarefa visa descobrir arestas que virão a existir na rede complexa quando esta evoluir (crescimento do número de nós e arestas com o passar do tempo) e não uma aresta perdida na construção da rede complexa. Entretanto, este método não funciona para todos os tipos de redes complexas, já que o método não consegue detectar comunidades em redes complexas que não possuam grupos bem definidos. Há uma coleção de trabalhos que vem usando e desenvolvendo algoritmos na área de predição de ligações [Kashima et al., 2009, Hasan et al., 2006, Kunegis and Lommatzsch, 2009, Lu and Zhou, 2009, Acar et al., 2009].

Uma das dificuldades da predição de ligações é que as redes complexas tendem a ser esparsas. Para driblar esta dificuldade, outros modelos fazem uso não só de propriedades estruturais do grafo mas também de características relacionais baseadas nos atributos dos nós do grafo. Esta abordagem é mais conhecida na área de aprendizado relacional ou aprendizado multi-relacional, que tem por objetivo não só o uso da estrutura dos grafos, mas também a descrição dos mesmos por meio de uma base de dados relacional ou lógica relacional ou de primeira ordem. Assim, além das ligações entre as tuplas formando um grafo, também há características, isto é, informações, relacionadas aos nós do grafo [Getoor and Diehl, 2005, Hasan et al., 2006, Taskar et al., 2004, Popescul et al., 2003]. Entretanto, esta informação complementar dos

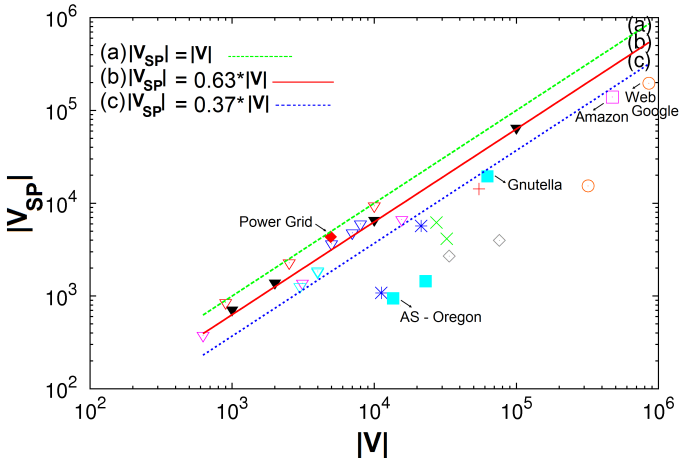


Figura 5.16. O padrão *NodeShatteringRatio* para redes reais e sintéticas (triângulos). A linha (a) ($\mathscr{V}_{sp} = \mathscr{V}$) mostra o valor de nós atual da rede igual ao valor original. A linha (b) mostra o valor de ($\mathscr{V}_{sp} = 0.37 * \mathscr{V}$) nós atual igual a 63% do valor de nós original, esta é conhecida como a quantidade de nós na fase de transição dos grafos *Erdös-Rényi* representados pelos triângulos pretos. A linha (c) ($\mathscr{V}_{sp} = 0.37 * \mathscr{V}$) o número de nós atual é 37% do número de nós total da rede. Isto comprova que as redes reais são bem resistentes e que a fase de transição esta bem distante.

nós e arestas nem sempre estão disponíveis, o que inviabiliza a aplicação desses algoritmos nesses casos.

5.4. Conclusão

Este documento apresentou uma visão geral da área de mineração de grafos e redes complexas. Esta área tem se mostrado muito importante atualmente, principalmente pelo grande crescimento de domínios de aplicação, nos quais os dados podem ser modelados através de redes complexas. Nestas aplicações, quando a modelagem é feita através técnicas tradicionais de representação e algoritmos tradicionais de mineração de dados (tais como regras de associação, detecção de agrupamentos, classificação, entre outros) são aplicados, os resultados tendem a não capturar todos os padrões relevantes (e presentes nos dados). Já as técnicas de mineração de grafos aplicadas a redes complexas podem trazer ganho substancial nos resultados da mineração. O documento apresentou ainda os principais algoritmos de mineração de grafos e propriedades estatísticas utilizados na área de redes complexas. Definições da teoria dos grafos necessárias para o entendimento desses algoritmos e propriedades básicas também foram abordadas. As redes complexas estão cada vez mais presentes nos sistemas computacionais e com isso o seu entendimento torna-se cada vez mais importante e relevante tanto para pesquisadores da área da computação quanto para de áreas em que problemas reais podem ser representadas através destes modelos.

Referências

[Abello et al., 1998] Abello, J., Buchsbaum, A. L., and Westbrook, J. R. (1998). A functional approach to external graph algorithms. In *Algorithmica*, pages 332–343.

Springer-Verlag.

[Acar et al., 2009] Acar, E., Dunlavy, D. M., and Kolda, T. G. (2009). Link prediction on evolving data using matrix and tensor factorizations. In Saygin, Y., Yu, J. X., Kargupta, H., Wang, W., Ranka, S., Yu, P. S., and Wu, X., editors, *ICDM Workshops*, pages 262–269. IEEE Computer Society.

[Albert et al., 1999] Albert, R., Jeong, H., and Barabasi, A.-L. (1999). The diameter of the world wide web.

[Albert et al., 2000] Albert, R., Jeong, H., and Barabási, A.-L. (2000). Error and attack tolerance of complex networks. *Nature*, 406:378–381.

[Alon, 1998] Alon, N. (1998). Spectral techniques in graph algorithms. In Lucchesi, C. L. and Moura, A. V., editors, *Lecture Notes in Computer Science 1380*, pages 206–215. Springer-Verlag, Berlin.

[Appel et al., 2009] Appel, A. P., Chakrabarti, D., Faloutsos, C., Kumar, R., Leskove, J., and Tomkins, A. (2009). Shatterplots: a fast tool for mining large graphs. In *SIAM SDM*, pages 802–813. SIAM.

[Bar-Yossef et al., 2002] Bar-Yossef, Z., Kumar, R., and Sivakumar, D. (2002). Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA*.

[Barabasi and Albert, 1999] Barabasi, A. L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

[Becchetti et al., 2008] Becchetti, L., Boldi, P., Castillo, C., and Gionis, A. (2008). Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *KDD*, pages 16–24.

[Bi et al., 2001] Bi, Z., Faloutsos, C., and Korn, F. (2001). The "DGX"distribution for mining massive, skewed data. *KDD*.

[Bollobás and Riordan, 2004] Bollobás, B. and Riordan, O. (2004). The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34.

[Bondy and Murty, 1979] Bondy, J. A. and Murty, U. S. R. (1979). *Graph Theory with applications*. Elsevier Science Publishing Co., Inc.

[Borgatti and Everett, 1999] Borgatti, S. P. and Everett, M. G. (1999). Models of core/periphery structures. *Social Networks*, 21.

[Broder et al., 2000] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. (2000). Graph structure in the web: experiments and models. In *Proceedings of the Ninth International World-Wide Web Conference (WWW9, Amsterdam, May 15 - 19, 2000 - Best Paper)*. Foretec Seminars, Inc. (of CD-ROM), Reston, VA.

[Chakrabarti et al., 2004a] Chakrabarti, D., Papadimitriou, S., Modha, D. S., and Faloutsos, C. (2004a). Fully automatic cross-associations. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 79–88. ACM Press.

[Chakrabarti et al., 2008] Chakrabarti, D., Wang, Y., Wang, C., Leskovec, J., and Faloutsos, C. (2008). Epidemic thresholds in real networks. *ACM Trans. Inf. Syst. Secur.*, 10(4):1–26.

[Chakrabarti et al., 2004b] Chakrabarti, D., Zhan, Y., and Faloutsos, C. (2004b). R-mat: A recursive model for graph mining. In *Fourth SIAM International Conference on Data Mining*.

[Chung et al., 2002] Chung, F., Chung, F., Chung, F., Lu, L., and Lu, L. (2002). The average distances in random graphs with given expected degrees. *Internet Mathematics*, 1:15879–15882.

[Chung and Lu, 2006] Chung, F. and Lu, L. (2006). *Complex Graphs and Networks*. American Mathematical Society.

[Chung, 1994] Chung, F. R. K. (1994). *Spectral Graph Theory*.

[Clauset et al., 2008] Clauset, A., Moore, C., and Newman, M. E. J. (2008). Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101.

[Clauset et al., 2004] Clauset, A., Newman, M. E. J., and Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70:066111.

[Clauset et al., 2007] Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2007). Power-law distributions in empirical data.

[Clauset et al., 2009] Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2009). Power-law distributions in empirical data. *SIAM Review*, 51(4):661–704.

[Cohen et al., 2000] Cohen, R., Erez, K., ben Avraham, D., and Havlin, S. (2000). Resilience of the internet to random breakdowns. *Phys. Rev. Lett.*, 85(21):4626–4628.

[Diestel, 2005] Diestel, R. (2005). *Graph Theory*. Springer-Verlag Heidelbergh.

[Dunbar, 1998] Dunbar, R. (1998). *Grooming, Gossip, and the Evolution of Language*. Harvard Univ Press.

[Faloutsos et al., 1999] Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On power-law relationships of the internet topology. In *SIGCOMM 1999*, volume 1, pages 251–262, Cambridge, Massachusetts. ACM Press.

[Flaxman et al., 2005] Flaxman, A., Frieze, A., and Fenner, T. (2005). High degree vertices and eigenvalues in the preferential attachment graph. *Internet Math.*, 2(1):1–19.

[Fortunato, 2010] Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3-5):75–174.

[Getoor and Diehl, 2005] Getoor, L. and Diehl, C. P. (2005). Introduction to the special issue on link mining. *SIGKDD Explor. Newsl.*, 7(2):1–2.

[Girvan and Newman, 2002] Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. In *Proc. Natl. Acad. Sci. USA*, volume 99.

[Guimerà et al., 2007] Guimerà, R., Sales-Pardo, M., and Amaral, L. A. (2007). Module identification in bipartite and directed networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 76(3 Pt 2).

[Hasan et al., 2006] Hasan, M. A., Chaoji, V., Salem, S., and Zaki, M. (2006). Link prediction using supervised learning. In *In Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security*.

[Holme, 2005] Holme, P. (2005). Core-periphery organization of complex networks. *Phys. Rev. E*, 72(4):046111.

[Huang, 2006] Huang, Z. (2006). Link prediction based on graph topology: The predictive value of the generalized clustering coefficient. In *Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (LinkKDD2006)*.

[Huberman and Adamic, 1999] Huberman, B. A. and Adamic, L. A. (1999). Internet: Growth dynamics of the world-wide web. *Nature*, 401(6749):131.

[Jeong et al., 2000] Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., and Barabási, A.-L. (2000). The large-scale organization of metabolic networks. *Nature*, 407.

[Karypis and Kumar, 1998] Karypis, G. and Kumar, V. (1998). Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel Distrib. Comput.*, 48:96–129.

[Kashima et al., 2009] Kashima, H., Kato, T., Yamanishi, Y., Sugiyama, M., and Tsuda, K. (2009). Link propagation: A fast semi-supervised learning algorithm for link prediction. In *SDM*, pages 1099–1110. SIAM.

[Kleinberg et al., 1999] Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. S. (1999). The web as a graph: measurements, models, and methods.

[Krishnamurthy and Wang, 2000] Krishnamurthy, B. and Wang, J. (2000). On network-aware clustering of web clients. *SIGCOMM Comput. Commun. Rev.*, 30(4):97–110.

[Kumar et al., 1999] Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. (1999). Trawling the web for emerging cyber-communities. *Comput. Networks*, 31(11-16):1481–1493.

[Kunegis and Lommatzsch, 2009] Kunegis, J. and Lommatzsch, A. (2009). Learning spectral graph transformations for link prediction. In *Proc. Int. Conf. in Machine Learning*.

[Latapy, 2008] Latapy, M. (2008). Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theor. Comput. Sci.*, 407(1-3):458–473.

[Leskovec et al., 2005] Leskovec, J., Kleinberg, J., and Faloutsos, C. (2005). Graphs over time: densification laws, shrinking diameters and possible explanations. In *eleventh ACM SIGKDD*, pages 177–187, New York, NY, USA. ACM Press.

[Leskovec et al., 2007] Leskovec, J., Kleinberg, J. M., and Faloutsos, C. (2007). Graph evolution: Densification and shrinking diameters. *ACM TKDD*, 1(1):1 – 40.

[Leskovec et al., 2008] Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. (2008). Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *CoRR*, abs/0810.1355.

[Liben-Nowell and Kleinberg, 2003] Liben-Nowell, D. and Kleinberg, J. (2003). The link prediction problem for social networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, New York, NY, USA. ACM.

[Lu and Zhou, 2009] Lu, L. and Zhou, T. (2009). Role of weak ties in link prediction of complex networks. In *Proceedings of the 1st ACM International Workshop on Complex Networks in Information and Knowledge Management (CNIKM)*, Hong Kong, China.

[McGlohon et al., 2008] McGlohon, M., Akoglu, L., and Faloutsos, C. (2008). Weighted graphs and disconnected components: patterns and a generator. In *KDD*, pages 524–532.

[Mihail and Papadimitriou, 2002] Mihail, M. and Papadimitriou, C. H. (2002). On the eigenvalue power law. In *RANDOM '02: Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 254–262, London, UK. Springer-Verlag.

[Milgram, 1967] Milgram, S. (1967). The small world problem. *Psychology Today*, 2:60–67.

[Newman, 2003] Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, 45:167–256.

[Newman, 2005] Newman, M. E. J. (2005). Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46:323.

[Newman, 2006] Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582.

[Nicoletti, 2006] Nicoletti, Maria Do Carmo ; Hruschka Jr., E. (2006). *Fundamentos da Teoria dos Grafos.*, volume 1. EdUFSCar - Editora da Universidade Federal de São Carlos, 1. ed. revisada edition.

[Page et al., 1998] Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library.

[Palmer et al., 2002] Palmer, C. R., Gibbons, P. B., and Faloutsos, C. (2002). Anf: A fast and scalable tool for data mining in massive graphs. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 1, pages 81–90, Edmonton, Alberta, Canada. ACM Press.

[Pennock et al., 2002] Pennock, D. M., Flake, G. W., Lawrence, S., Glover, E. J., and Giles, C. L. (2002). Winners don't take all: Characterizing the competition for links on the Web. *Proceedings of the National Academy of Sciences*, 99(8):5207–5211.

[Popescul et al., 2003] Popescul, A., Popescul, R., and Ungar, L. H. (2003). Statistical relational learning for link prediction.

[Radicchi et al., 2004] Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., and Parisi, D. (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663.

[Ravasz et al., 2002] Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., and Barabasi, A. L. (2002). Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555.

[Reddy et al., 2002] Reddy, P. K., Kitsuregawa, M., Sreekanth, P., and 0002, S. S. R. (2002). A graph based approach to extract a neighborhood customer community for collaborative filtering. In *DNIS*, pages 188–200.

[Redner, 1998] Redner, S. (1998). How popular is your paper? an empirical study of the citation distribution.

[Reka and Barabási, 2002] Reka, A. and Barabási (2002). Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97.

[Ripeanu et al., 2002] Ripeanu, M., Foster, I., and Iamnitchi, A. (2002). Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1).

[Schroeder, 1991] Schroeder, M. (1991). *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*. W.H. Freeman and Company, New York.

[Siganos, 2006] Siganos, Georgos; Sudhir L Tauro, M. F. (2006). Jellyfish: A conceptual model for the as internet topology. In *Journal of Communications and Networks*, volume 8, pages 339 – 350.

[Taskar et al., 2004] Taskar, B., Wong, M., Abbeel, P., and Koller, D. (2004). Link prediction in relational data.

[Tauro et al., 2001] Tauro, S. L., Palmer, C., Siganos, G., and Faloutsos, M. (2001). A simple conceptual model for the internet topology. In *Global Internet, San Antonio, Texas*.

[Tsourakakis, 2008] Tsourakakis, C. E. (2008). Fast counting of triangles in large real networks without counting: Algorithms and laws. In *ICDM '08*, pages 608–617, Washington, DC, USA. IEEE Computer Society.

[Watts and Strogatz, 1998] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393(6684):440–442.

[Wu et al., 2007] Wu, J., Zhang, Y., Mao, Z. M., and Shin, K. G. (2007). Internet routing resilience to failures: analysis and implications. In *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, New York, NY, USA. ACM.

Capítulo 6

Processo de KDD aplicado à Bioinformática

Ana T. Winck, Karina S. Machado, Duncan D. Ruiz e Osmar Norberto de Souza

Resumo

A computação tem contado com pesquisas interdisciplinares, como a bioinformática, que tem a característica de possuir grandes volumes de dados. O gerenciamento e a análise desse tipo de dados aparecem como importantes campos de investigação. A construção de um ambiente de descoberta de conhecimento em base de dados (KDD – Knowledge Discovery in Databases) pode ser vista como uma abordagem que pode apresentar importantes desafios se direcionado para o tratamento de dados tão peculiares e de característica não convencional, como na bioinformática. Propõe-se apresentar e discutir todas as etapas que compõe o processo de KDD, e como tal processo pode ser aplicado à bioinformática. Acredita-se que as lições aprendidas possam ser úteis no desenvolvimento de futuras pesquisas interdisciplinares.

Abstract

The computer science has relied on interdisciplinary researches, as bioinformatics, which usually posses a large amount of data. The management and analysis of such kind of data appear as an important research field. To build a KDD (Knwoledge Discovery in Databases) environment can be viewed as an approach that present important challenges if directed to the treatment of these particular and non-conventional data, as in bioinformatics. We propose to present and discuss all KDD steps, and how such process can be applied on bioinformatics. We believe that the lessons learned can be useful to the development of future interdisciplinary researches.

6.1. Introdução

A computação, nas suas diferentes áreas de aplicação, vem sendo enriquecida com pesquisas interdisciplinares, isto é, pesquisas que fazem uso da computação para atender a outras áreas de conhecimento. Dentre as diferentes áreas que se beneficiam com essa interação, pode-se citar a bioinformática. Para Wang et al. (2005), bioinformática é a

ciência de gerenciar, minerar, integrar e interpretar informações a partir de dados biológicos, enfatizando o constante crescimento do número de dados que demandam por essas análises. A bioinformática consolidou-se quando da implantação do projeto GENOMA: os seqüenciadores automáticos de DNA produziram uma grande quantidade de sequências, tornando-se necessário o investimento em recursos específicos de armazenamento e análise desses dados biológicos (Luscombe et al. 2001). A partir daí surgiram várias áreas de atuação em bioinformática. Lesk (2002) aponta algumas principais frentes, como genomas, proteomas, alinhamento de árvores filogenéticas, biologia de sistemas, estruturas de proteínas e descoberta de fármacos.

Do ponto de vista da computação, diversas áreas de atuação podem ser empregadas para atender a essa demanda. Dentre elas, destaca-se o processo de descoberta de conhecimento em banco de dados (KDD – *Knowledge Discovery in Databases*), proposto por Fayyad et al. (1996). Este processo, difundido e conhecido pela comunidade de banco de dados, pode apresentar importantes desafios se direcionado para o tratamento de dados tão peculiares e de característica não-convencional, como na bioinformática.

Uma área de pesquisa que tem sido investigada em bioinformática é o desenho racional de fármacos (RDD – *Rational Drug Design*) (Kuntz, 1992). O princípio fundamental do RDD é a interação entre receptores e ligantes (Lybrand, 1995). Ligantes são definidos como moléculas que se ligam a outras moléculas biológicas, chamadas receptores, para realizar ou inibir funções específicas (Balakin, 2009). É na docagem molecular que se investiga e avalia o melhor encaixe do ligante no receptor (Kuntz, 1992). Um dos maiores desafios dessa área de pesquisa é lidar com o grande volume de dados envolvidos: catalogação de ligantes, conformações do receptor obtidas por simulações pela dinâmica molecular (DM) e resultados de experimentos de docagem molecular. Este minicurso propõe-se a apresentar e discutir todas as etapas que compõem o processo de KDD e como esse processo pode ser aplicado à bioinformática no contexto de RDD.

O restante deste trabalho está organizado conforme segue. A seção 6.2 apresenta uma introdução à bionformática, apresentando os principais termos envolvidos. A seção 6.3 relata os passos de construção de um típico processo de KDD. Na seção 6.4 são apresentados os exemplos práticos de construção de um processo de KDD para a bioinformática.

6.2. Introdução à bioinformática

Inicialmente, bioinformática era definida como uma área interdisciplinar envolvendo biologia, ciência da computação, matemática e estatística para analisar dados biológicos (Mount, 2004). Com o advento da era genômica, bioinformática passou a ser definida como biologia em termos de moléculas e a aplicação da computação para entender e organizar a informação associada com esses dados biológicos em larga escala (Luscombe et al., 2001).

Para Lesk (2002) uma das principais características dos dados de bioinformática é o seu grande volume. Como exemplo, tem-se o banco de dados de sequências de nucleotídeos GenBank (Benson et al, 2008), que até agosto de 2009 já continha depositadas 106.533.156.756 bases em 108,431,692 seqüências. O *GenBank* e outros bancos de dados biológicos não são apensas extensos, mas crescem a taxas bastante

elevadas (Lesk, 2002). Esse grande volume de dados biológicos e seu crescimento elevado definem os principais objetivos da bioinformática: organizar os dados biológicos de maneira que os pesquisadores possam acessar informações já cadastradas, assim como submeter novas entradas, na medida em que vão sendo produzidas; desenvolver ferramentas e pesquisas que ajudem na análise dos dados e utilizar as ferramentas desenvolvidas para interpretar os dados de forma que tenham significado biológico (Luscombe et al., 2001).

Hunter (1993) diz que um dos maiores desafios dos cientistas da computação que pretendem trabalhar na área de biologia molecular consiste em familiarizar-se com o complexo conhecimento biológico existente e seu vocabulário técnico extenso. Sendo assim, a seguir serão revisados conceitos necessários para um entendimento básico sobre os dados envolvidos neste trabalho.

6.2.1. O dogma central da biologia molecular

O DNA é responsável por codificar todas as proteínas. Esse processo, definido como o dogma central da biologia molecular, compreende a fase de transcrição, onde o DNA é processado, gerando o mRNA, ou RNA mensageiro, que é enviado para fora do núcleo da célula, onde a mensagem é traduzida em proteínas.

6.2.2. Proteínas

Proteínas são as macromoléculas mais abundantes nas células vivas e se encontram em todas as células e em todas as partes das células. Elas realizam a maioria das funções catalisando as reações químicas nas células vivas. A estrutura das diferentes proteínas é construída com o conjunto dos 20 aminoácidos básicos.

Todos os aminoácidos têm uma estrutura conforme Figura 6.1a, contendo um grupo carboxila (COOH), um grupo amino (NH2), um átomo de hidrogênio, ligados ao mesmo átomo de carbono (C). A diferenciação entre os 20 aminoácidos ocorre devido à presença de um radical R, que confere aos aminoácidos uma série de características como, por exemplo, polaridade e grau de ionização. A Figura 6.1b mostra um exemplo de aminoácido, a alanina, cujo radical é CH3.

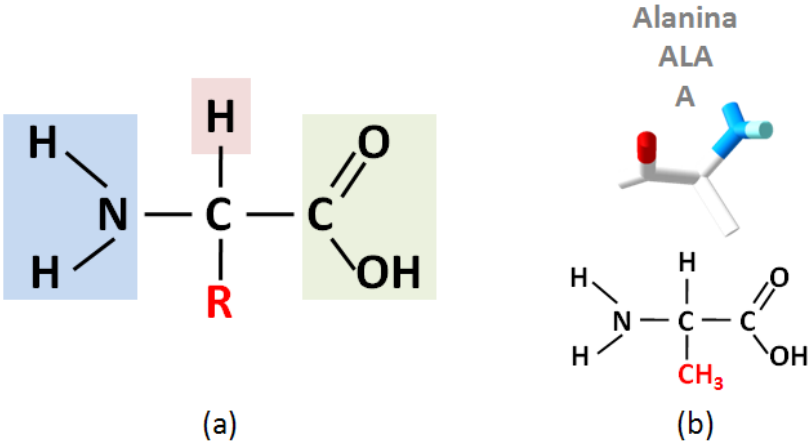


Figura 6.1. (a) Estrutura de um aminoácido. (b) Estrutura do aminoácido Alanina.

De acordo com Hunter (1993), as cadeias de aminoácidos são formadas por uma reação que ocorre entre o átomo de nitrogênio no final do grupo amina de um aminoácido e o átomo de carbono do grupo carboxila de outro, ligando os dois aminoácidos e liberando uma molécula de água. A ligação é chamada de ligação peptídica, e longas cadeias de aminoácidos formam os chamados polipeptídeos. Todas as proteínas são polipeptídeos, apesar deste termo geralmente referir-se a cadeias menores que proteínas inteiras.

A seqüência contínua de aminoácidos de uma proteína é chamada de estrutura primária (Figura 6.2a), e é o nível mais básico de organização de uma proteína. Os aminoácidos da estrutura primária podem adquirir conformações secundárias como α-hélice ou folhas-β (Figura 6.2b). As ligações de hidrogênio entre os aminoácidos que formam a proteína favorecem essas estruturas secundárias. A composição desses elementos estruturais formando um ou mais domínios estabelecem a estrutura terciária de uma proteína (Figura 6.2c). A estrutura final, estrutura quaternária, pode conter várias cadeias polipeptídicas (Figura 6.2d). As formações destas estruturas terciárias e quaternárias permitem o surgimento de regiões funcionais chamadas sítio ativo (que confere atividade biológica à proteína).

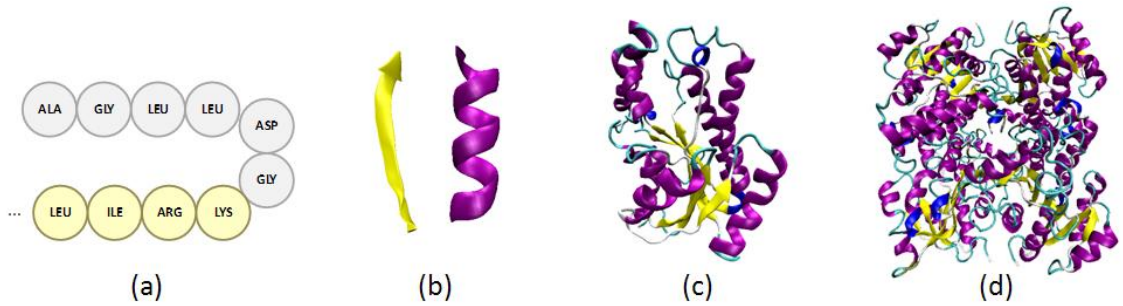


Figura 6.2. (a) Estrutura primária de uma proteína. (b) Estruturas secundárias de uma proteína: folha-β e α-hélice. (c) Exemplo de estrutura terciária de uma proteína (PDBCode:1ENY). (d) Exemplo de estrutura quaternária de uma proteína (PDBCode:1BVR).

6.2.3. Principais áreas de atuação em bioinformática

Os dados de DNA, RNA e proteínas podem ser utilizados em diferentes áreas de bioinformática, definindo algumas das principais atividades na área: análise de seqüências de proteínas, alinhamento múltiplo de seqüências, visualização de estrutura de proteínas, predição de estrutura de proteínas, ferramentas para genômica e proteômica, bancos de dados biológicos, desenho racional de fármacos, entre outros. Entre as atividades descritas, este trabalho está inserido no contexto de desenho racional de fármacos, ou *Rational Drug Design* (RDD) (Kuntz, 1992). Esse contexto e os processos envolvidos no mesmo serão descritos a seguir.

6.2.4. Desenho racional de fármacos

Segundo Drews (2003), com o avanço da biologia molecular e das técnicas de simulação por computador, o planejamento de medicamentos passou a ser feito de

maneira mais lógica, o que é chamado de desenho racional de drogas. O processo de RDD é composto de quatro etapas (Kuntz, 1992):

1. A primeira etapa consiste em isolar um alvo específico, chamado receptor, (proteínas, receptores de membrana, DNA, RNA, etc.). A partir da análise computacional da estrutura tridimensional (3D) dessa proteína armazenada em um banco de dados estrutural como o *Protein Data Bank* (PDB) (Berman et al., 2000), é possível apontar prováveis regiões de ligação, por exemplo, regiões onde uma pequena molécula, chamada ligante, pode se ligar a esse receptor;
2. Baseado nas prováveis regiões de ligação identificadas na etapa anterior é selecionado um conjunto de prováveis candidatos a ligantes que podem se ligar a essa região no receptor. As diferentes conformações que um dado ligante pode assumir dentro do sítio de ligação de uma determinada proteína podem ser simuladas por software de docagem molecular como AutoDock3.0.5 (Morris et al. 1998);
3. Os ligantes que teoricamente obtiveram melhores resultados nas simulações são experimentalmente sintetizados e testados;
4. Baseado nos resultados experimentais, o medicamento é gerado ou o processo retorna à etapa 1 com pequenas modificações no ligante.

Sendo assim, a interação entre moléculas é o princípio do desenho de drogas, sendo um dos mais interessantes desafios nessa área (Broughton, 2000). É na docagem molecular que é procurado o melhor encaixe do ligante na estrutura alvo ou receptor.

6.2.5. Docagem molecular e simulações pela dinâmica molecular

De acordo com Lybrand (1995), um medicamento deve interagir com um receptor para exercer uma função fisiológica vinculada à ligação dessa com outras moléculas, e essas ligações determinam se as funções serão estimuladas ou inibidas. Essas ligações ocorrem em locais específicos, chamados sítios ativos ou de ligação. A associação entre duas moléculas no sítio de ligação não depende somente do encaixe. Existe a necessidade de haver uma energia favorável para que essa interação ocorra. Essa energia é determinada pela carga e tamanho dos átomos ali contidos. Estas ligações que ocorrem entre os átomos são medidas pela quantidade de energia despendida, quanto mais negativa, melhor a interação entre as moléculas.

Esse processo de ligar uma pequena molécula a uma proteína-alvo não é um processo simples: muitos fatores entrópicos e entálpicos influenciam as interações entre eles. A mobilidade do ligante e do receptor, o efeito do ambiente no receptor (proteína), a distribuição de carga no ligante, e outras interações dos mesmos com a água, complicam muito a descrição desse processo. A maioria dos algoritmos que executam docagem molecular somente considera a flexibilidade do ligante, considerando o receptor rígido ou as possíveis orientações das cadeias laterais são selecionadas baseadas em uma visão subjetiva. Entretanto, sabe-se que as proteínas não permanecem rígidas em seu ambiente celular, sendo de fundamental importância a consideração dessa flexibilidade do receptor na execução dos experimentos de docagem molecular.

Há muitos trabalhos sendo desenvolvidos para a incorporação da flexibilidade de receptores na docagem molecular, como revisado por Totrov e Abagyan (2008) e Chandrika et al. (2009). Entre as várias abordagens disponíveis, neste trabalho estamos considerando a execução de uma série de experimentos de docagem molecular, considerando em cada experimento uma conformação do receptor gerada por uma simulação pela dinâmica molecular (DM) (Lin et al. 2002; Machado et al., 2007, Amaro et al., 2008). A simulação pela DM é uma das técnicas computacionais mais versáteis e amplamente utilizadas para o estudo de macromoléculas biológicas (van Gunsteren 1990). Com simulações pela DM é possível estudar o efeito explícito de ligantes na estrutura e estabilidade das proteínas, os diferentes parâmetros termodinâmicos envolvidos, incluindo energias de interação e entropias.

O principal problema com a utilização da trajetória da DM em docagem molecular é o tempo necessário para a execução de experimentos e a grande quantidade de dados gerados. Visando reduzir esse tempo de execução e melhor entender como ocorre a interação receptor-ligante considerando a flexibilidade do receptor, nesse trabalho está sendo aplicado um processo de KDD. Para alcançar esse objetivo é necessário um conjunto de etapas que serão descritas nas próximas seções.

6.3. Processo de KDD

O processo KDD (Fayyad et. al, 1996; Han e Kamber, 2006) é uma sequência de etapas interativas e iterativas direcionadas à tomada de decisão, especialmente quando da existência de um grande volume dados. A Figura 6.3 ilustra uma adaptação da visão de Han e Kamber (2006) quanto às diferentes etapas do processo de KDD. Por essa figura nota-se que um processo de KDD inicia-se a partir da existência de um grande volume de dados (a) que merecem ser analisados. Como esse volume de dados, na maioria das vezes, é proveniente de fontes heterogêneas, os mesmos passam por uma etapa de transformação (b) para que sejam representados em um único padrão de referência. Uma vez transformados, os dados são devidamente armazenados em uma base de dados alvo (c), muitas vezes projetada na forma de um *data warehouse* (DW). Os dados devidamente organizados na base de dados alvo fornecem suporte para diferentes tipos de análises, onde a principal técnica de análise presente no processo de KDD é a mineração de dados (e). Para que os dados sejam minerados, é importante que passem por um processo de pré-processamento (d), etapa importante para que algoritmos de mineração produzam melhores resultados. Os modelos induzidos a partir da mineração apresentam padrões (f) e, após analisados, podem atingir o conhecimento (g) esperado.

Embora cada etapa que compõe o processo de KDD seja suficientemente abrangente para ser tratada de forma independente, existe uma forte relação de dependência entre elas. Para melhor falar a respeito delas, iremos dividi-las em duas principais etapas:

- A construção de uma base de dados alvo – abrangendo as etapas (a), (b) e (c) da Figura 6.3; e
- Etapas de mineração de dados – a qual é composta pelas etapas (d), (e), (f) e (g) da Figura 6.3.

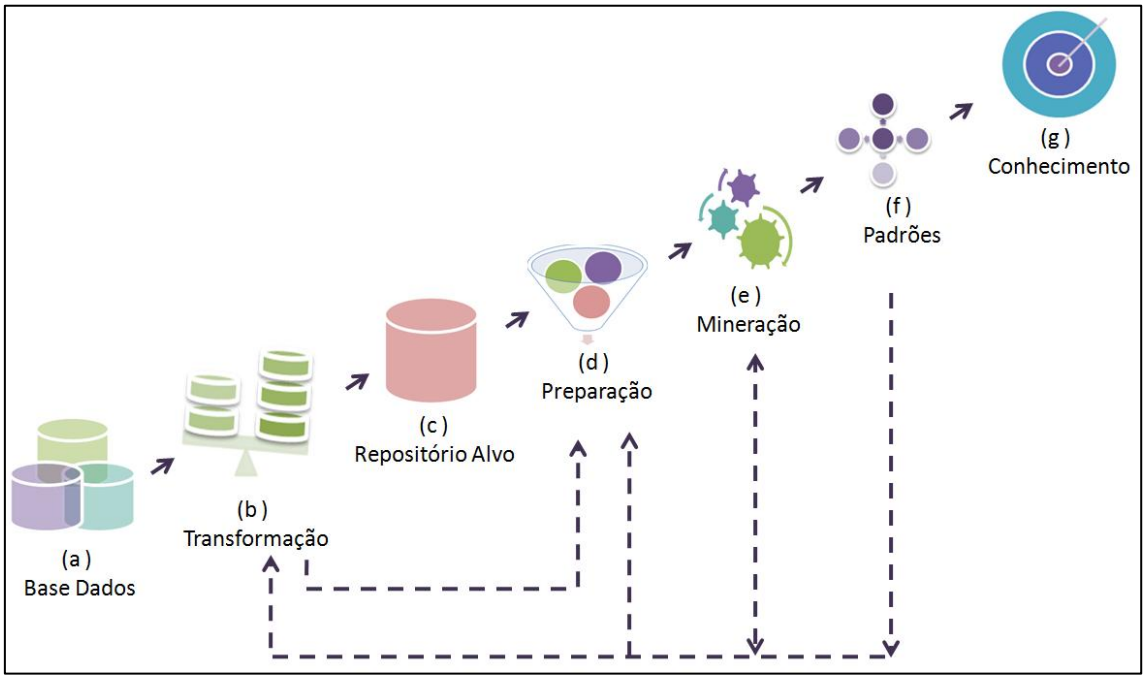


Figura 6.3. Processo de KDD adaptado de Han e Kamber (2006).

6.3.1. Construção de uma base de dados alvo

Um processo de KDD é comumente construído quando há interesse analítico sobre distintas fontes de dados, sendo que essas diferentes fontes muitas vezes têm características heterogêneas. A idéia é que esses dados sejam armazenados em um repositório alvo, semanticamente consistente. Nesse sentido, a heterogeneidade dos dados de origem é tratada de forma a garantir a integração dos dados em um único formato.

Han e Kamber (2006) sugerem que esse repositório alvo seja construído na forma de um DW, buscando manter um histórico organizado dos registros, de modo a auxiliar a tomada de decisão. Inmon (1997) apresenta a clássica definição de um DW como sendo um conjunto de dados baseado em assuntos, integrado, não-volátil, e variável em relação ao tempo, de apoio às decisões estratégicas. DW são construídos de forma a satisfazer sua estrutura multidimensional (Kimball e Ross, 2002). Um modelo analítico, que representa essa estrutura multidimensional, possui dois tipos de tabelas: fato e dimensão. Entende-se por dimensão as perspectivas de uma base de dados que possam gerar registros referentes às características do problema modelado. Essas são, tipicamente, organizadas em torno de um termo central, ou tabela fato, a qual contém os atributos chave das dimensões e atributos que representem valores relevantes ao problema.

Um dos processos mais importantes para a construção de uma base de dados alvo é a etapa de ETC – extração, transformação e carga. A extração é a primeira fase, a qual visa percorrer as distintas bases de dados e capturar apenas os dados significantes ao domínio. Na fase de transformação, os dados capturados podem sofrer algum tipo de tratamento, ajustando possíveis conflitos de tipificação, e eliminando possíveis ruídos ou conteúdos irrelevantes ao repositório alvo. Por fim, é feita a devida carga no

repositório (Kimball e Ross, 2002). No processo de KDD representado pela Figura 6.3, a fase de ETC está presente na etapa de transformação, Figura 6.3(b).

Tendo o repositório alvo modelado, desenvolvido e os dados devidamente armazenados, já é possível efetuar consultas analíticas sobre esses dados. Um exemplo, para o caso de um DW construído, é a manipulação de seus dados por ferramentas OLAP (*On-Line Analytical Processing*). Operações como pivoteamento de dados, *roll-up*, *drill-down* e *slice&dice*, são operações típicas de ambientes de processamento analítico de dados. Além dessa abordagem, entretanto, e seguindo as demais etapas do processo de KDD, os dados estão aptos a serem trabalhados pela etapa de mineração de dados.

6.3.2. Etapas de mineração de dados

A mineração de dados é a etapa do processo de KDD que converte dados brutos em informação. A mineração de dados divide-se em diferentes tarefas: descritivas e preditivas. As técnicas descritivas sumarizam relações entre dados, tendo como objetivo aumentar o entendimento a respeito deles. As técnicas preditivas têm o objetivo de apontar conclusões quanto aos dados analisados, prevendo valores para um dado atributo de interesse. Dentre as tarefas descritivas pode-se citar como exemplo regras de associação e agrupamento. Já algoritmos de classificação e regressão podem ser vistos como exemplos de tarefas preditivas (Tan et al., 2006). Este trabalho concentra-se em regras de associação, para tarefas descritivas, e árvores de decisão para classificação e regressão, para tarefas preditivas.

Regras de associação são implicações na forma $x \rightarrow y$. Em aprendizagem de máquina, regras de associação possuem duas principais medidas de qualidade: suporte e confiança. A primeira diz respeito à significância de uma regra e a segunda se refere ao percentual de ocorrências em que os itens de x ocorrem simultaneamente (Alpaydim, 2004).

Tarefas preditivas são compostas, basicamente, de uma entrada x e um resultado de saída y, onde a tarefa é aprender como mapear a entrada para a o resultado de saída. Esse mapeamento pode ser definido como uma função $y = g(x|\theta)$ onde $g(.)$ é o modelo e θ são seus respectivos parâmetros (Alpaydim, 2004). Embora existam vários algoritmos que fazem uso de tarefas preditivas, muitos deles apenas constroem uma função que indica um dado atributo alvo a que os objetos pertencem. Entretanto, em muitos problemas de mineração é necessário compreender o modelo induzido. Segundo Freitas et al. (2010), apesar da falta de consenso na literatura de mineração de dados sobre as tarefas que apresentam resultados mais compreensíveis, existe um acordo razoável de que representações como árvores de decisão são melhor compreendidas pelos usuários do que algoritmos de representação caixa-preta. Ainda segundo Freitas et al. (2010), árvores de decisão têm a vantagem de graficamente representar o conhecimento descoberto e sua estrutura hierárquica pode apontar a importância dos atributos utilizados para a predição.

Independentemente da tarefa e algoritmo de mineração sendo aplicados, é importante que os dados a serem minerados passem por uma criteriosa etapa de pré-processamento. Essa etapa, que não deve ser confundida com a etapa de ETC, é fundamental para que os modelos de mineração induzidos apresentem resultados mais satisfatórios. O pré-processamento dos dados faz uso dos dados que já estão inseridos na

base de dados alvo e adequá-los para o algoritmo de mineração o qual esses dados serão submetidos. Diversos autores (Tan et al., 2006; Han e Kamber, 2001; Fayyad et al., 1996) estão de acordo que um pré-processamento conveniente é um importante ponto a ser considerado.

6.4. Exemplos práticos

Neste trabalho propõe-se apresentar como desenvolver um processo de KDD em bioinformática, sobre bases de dados de docagem molecular. Esta seção é dividida em duas subseções. Em 6.4.1 são descritos os dados utilizados nesse trabalho, apresentando a coleta e tratamento dos mesmos, bem como a construção de uma base de dados alvo para seu armazenamento. Em 6.4.2 apresenta-se desafios e oportunidades de análise sobre esses dados, relatando o processo de pré-processamento dos mesmos até a utilização de algoritmos de mineração. Para esse trabalho é utilizado o ambiente WEKA (Hall et al., 2009) para a mineração de dados.

6.4.1. Dados biológicos utilizados neste trabalho

Para a execução de docagem molecular é necessário um receptor, um ligante e um software para executar as simulações. Nesse trabalho estamos considerando como receptor a enzyma InhA do *Mycobacterium tuberculosis* (Mtb) (Dessen et al, 1995); como ligante, a nicotinamida adenina dinucleotídeo (NADH) (Dessen et al, 1995); e como software de docagem o AutoDock3.0.5 (Morris et al., 1998).

6.4.1.1. Receptor e ligantes

A proteína InhA representa um importante alvo para o controle da tuberculose (Oliveira et al. 2007), pois ela inibe a biossíntese de ácidos micólicos, um importante componente da parede celular da micobactéria, e conseqüentemente uma das estruturas essenciais para a sobrevivência da mesma. A estrutura tridimensional dessa proteína está descrita na Figura 6.4, na forma de *ribbons*. Ao lado da estrutura, na mesma figura, é apresentado parte do arquivo PDB desta proteína. O arquivo PDB da InhA utilizado neste trabalho (PDBCode:1ENY) foi obtido do *Protein Data Bank* (Berman et al 2000). Um arquivo PDB é composto por um cabeçalho, que não está descrito na Figura 6.4, que descreve como a estrutura da proteína foi obtida (o método, a resolução, o artigo que descreve a estrutura, os autores, etc) e pela descrição dos átomos e resíduos que compõem essa estrutura juntamente com suas coordenadas cartesianas.

Estrutura - InhA

PDB - InhA

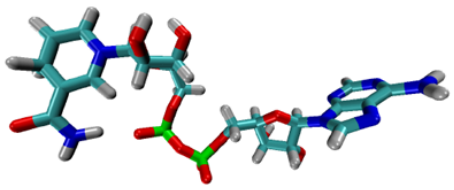
```
ATOM      1  N   ALA A   2       9.036  61.230  21.932  1.00 66.94
ATOM      2  CA  ALA A   2       8.304  60.436  22.930  1.00 65.47
ATOM      3  C   ALA A   2       8.723  58.949  23.009  1.00 63.24
ATOM      4  O   ALA A   2       9.847  58.587  23.415  1.00 63.92
ATOM      5  CB  ALA A   2       6.778  60.550  22.702  1.00 65.37
ATOM      6  N   GLY A   3       7.794  58.097  22.595  1.00 59.43
ATOM      7  CA  GLY A   3       7.959  56.651  22.672  1.00 54.68
ATOM      8  C   GLY A   3       6.751  56.322  23.547  1.00 49.10
ATOM      9  O   GLY A   3       6.798  56.505  24.763  1.00 49.91
ATOM     10  N   LEU A   4       5.621  56.069  22.913  1.00 42.86
ATOM     11  CA  LEU A   4       4.409  55.790  23.644  1.00 37.47
ATOM     12  C   LEU A   4       4.703  54.675  24.622  1.00 33.16
ATOM     13  O   LEU A   4       4.123  54.641  25.700  1.00 31.92
ATOM     14  CB  LEU A   4       3.333  55.321  22.649  1.00 35.56
ATOM     15  CG  LEU A   4       1.921  55.094  23.156  1.00 33.46
...
```

Figura 6.4. Estrutura tridimensional do receptor InhA e parte do arquivo PDB desta estrutura (PDBCode: 1ENY).

No caso da InhA, essa proteína é composta por 4.008 átomos (os números e nomes de alguns desses átomos estão descritos nas colunas 2 e 3 do PDB da Figura 6.4) distribuídos em 268 resíduos (o nome e o número de alguns resíduos estão descritos nas colunas 4 e 5 do PDB descrito na Figura 6.4).

O ligante NADH - Nicotinamida Adenina Dinucleotídeo, forma reduzida (Dessen et al, 1995) é o ligante natural da enzima InhA. Essa molécula tem um total de 71 átomos (após a preparação para a docagem molecular, permanece com um total de 52 átomos, pois perde os hidrogênios polares) e sua estrutura tridimensional está descrita na Figura 6.5, assim como parte do arquivo mol2 deste ligante utilizando nos experimentos de docagem molecular com o mesmo. O arquivo mol2 descreve as coordenadas dos átomos do ligante, a carga de cada átomo (na última coluna onde estão descritos os átomos) e como os átomos são conectados na estrutura do ligante (parte final do arquivo mol2).

Estrutura - NADH



Mol2 - NADH

```
@<TRIPOS>MOLECULE
UNTITLED
71 75 1 0 0
SMALL
USER_CHARGES


@<TRIPOS>ATOM
      1 C6N         -5.5959   -4.3594    1.0216 C.2       1     ****   -0.3550
      2 H6N         -5.0019   -4.4531    0.1243 H         1     ****    0.2220
      3 C5N         -6.1594   -5.4464    1.7507 C.2       1     ****   -0.1730
      4 H5N         -6.0147   -6.4642    1.4173 H         1     ****    0.1260
      5 C4N         -6.9471   -5.1565    2.8972 C.3       1     ****    0.1350
      6 H41         -7.9253   -5.5901    2.6881 H         1     ****    0.0200
...
    64   62   61  1
    65   63   62 ar
    66   64   63  1
    67   65   64  1
    68   66   64  1
    69   67   63 ar
    70   68   67 ar
    71   69   68  1
    72   70   68 ar
    73   71   58  1
    74   71   62 ar
    75   71   70 ar
@<TRIPOS>SUBSTRUCTURE
```

Figura 6.5. Estrutura tridimensional do ligante NADH e parte do arquivo mol2 desta estrutura (PDBCode: 1ENY).

6.4.1.2. Simulações pela DM do receptor InhA

Os estudos de simulação por DM desse receptor, que originaram as conformações utilizadas nesse trabalho, foram realizados com a InhA complexada com o NADH utilizando o software AMBER6.0 (Pearlman et al, 1995) por um período de 3.100 ps (picosegundos = 10^{-12} s) e estão descritos no trabalho de Schroeder et al. (2005). Cada conformação do receptor é um arquivo PDB, conforme descrito na Figura 6.4.

6.4.1.3. Docagem molecular com o receptor InhA e o ligante NADH

Para considerar o receptor flexível nas simulações de docagem molecular, utilizando o ligante NADH no seu formato mol2 (Figura 6.5), foram submetidos 3.100 experimentos de docagem molecular (onde, em cada experimento uma das 3.100 conformações do receptor foi considerada) utilizando o workflow científico descrito em Machado et al. (2007). Esse workflow foi desenvolvido para executar automaticamente este tipo de experimento de docagem molecular com o receptor flexível. O software de docagem molecular utilizado foi o AutoDock3.0.5 (Morris et al. 1998) e como protocolo de execução o algoritmo *Simulated Annealing* (SA) com 10 runs onde o ligante foi mantido rígido. Cada *run* corresponde a uma diferente tentativa de encontrar a melhor energia de

interação entre o receptor-ligante (*Free Energy of Binding* - FEB). Como resultado da execução de um experimento de docagem utilizando o Autodock3.0.5 tem-se um arquivo de saída descrito em parte na Figura 6.6, que lista os resultados da docagem (destacados na figura): as coordenadas finais do ligante (sua posição final após a docagem), a FEB, o valor do *Root Mean Squared Deviation* – RMSD que corresponde a distância do centro de massa do ligante em sua posição inicial e final após a docagem e outros valores para cada uma das 10 tentativas executadas, ordenadas ascendentemente por FEB.

```
Residue number will be set to the conformation's cluster rank.

MODEL        8
USER    Run = 8
USER    Cluster Rank = 1
USER    Number of conformations in this cluster = 1
USER
USER    RMSD from reference structure       = 5.197 A
USER
USER    Estimated Free Energy of Binding    =  -15.18 kcal/mol  [=(1)+(3)]
USER    Estimated Inhibition Constant, Ki   =   +7.52e-12       [Temperature = 298.15 K]
USER
USER    Final Docked Energy                 =  -17.10 kcal/mol  [=(1)+(2)]
USER
USER    (1) Final Intermolecular Energy     =  -17.04 kcal/mol
USER    (2) Final Internal Energy of Ligand =   -0.06 kcal/mol
USER    (3) Torsional Free Energy           =   +1.87 kcal/mol
USER
USER
USER    DPF = LIGmoved.MACRO_pdbqs.dpf
USER    NEWDPF move     LIGmoved.pdbq
USER    NEWDPF about    -4.837000 6.875000 0.114000
USER    NEWDPF tran0    -0.354736 7.653123 4.259508
USER    NEWDPF quat0    -0.510294 -0.778544 0.365334 -38.520748
USER    NEWDPF ndihe    6
USER    NEWDPF dihe0    104.50 157.90 61.05 22.51 -174.54 103.78
USER
USER                  Rank         x       y       z       vdW    Elec       q      RMS
ATOM      1  C5'A***     1      -0.355   7.653   4.260    -0.34  +0.04  +0.192   5.197
ATOM      2  C4'A***     1       0.951   8.331   3.948    -0.45  +0.06  +0.224   5.197
ATOM      3  O4'A***     1       0.876   9.415   2.950    -0.13  -0.01  -0.355   5.197
ATOM      4  C3'A***     1       1.606   8.966   5.195    -0.36  +0.05  +0.264   5.197
ATOM      5  O3'A***     1       2.759   8.146   5.512    +0.02  -0.16  -0.654   5.197
ATOM      6  HO3A***     1       3.344   8.137   4.752    +0.09  +0.11  +0.438   5.197
```

Figura 6.6. Parte do arquivo de saída do software AutoDock3.0.5.

6.4.1.4. Armazenamento dos dados

Como mencionado, uma das etapas do processo de KDD é o desenvolvimento de uma base de dados alvo. Sendo assim, todos os dados sobre o receptor e suas conformações geradas na simulação pela DM, dados sobre o ligante e dados sobre os resultados dos experimentos de docagem molecular, passaram por um processo de ETC que homogeneizou e integrou os dados das diferentes fontes onde os mesmos foram obtidos, e foram armazenados em uma base de dados chamada FReDD (Winck et al., 2009). O modelo desse banco de dados está na Figura 6.7. Atualmente esse banco de dados é composto por 17 tabelas contendo um total de 15.814.183 registros. Esses registros estão relacionados aos dados sobre o receptor InhA e suas 3.100 conformações, sobre o ligante NADH e mais 3 ligantes: Isoniazida Pentacianoferrato-IPF (Oliveira et al, 2004), Triclosano-TCL (Kuo et al, 2003) e Etionamida ou ETH (Wang et al., 2007) . Além disso, estão também armazenados todos os resultados de docagem molecular da InhA flexível e os 4 ligantes. Neste trabalho serão utilizados na etapa de mineração de dados somente os dados do complexo InhA-NADH.

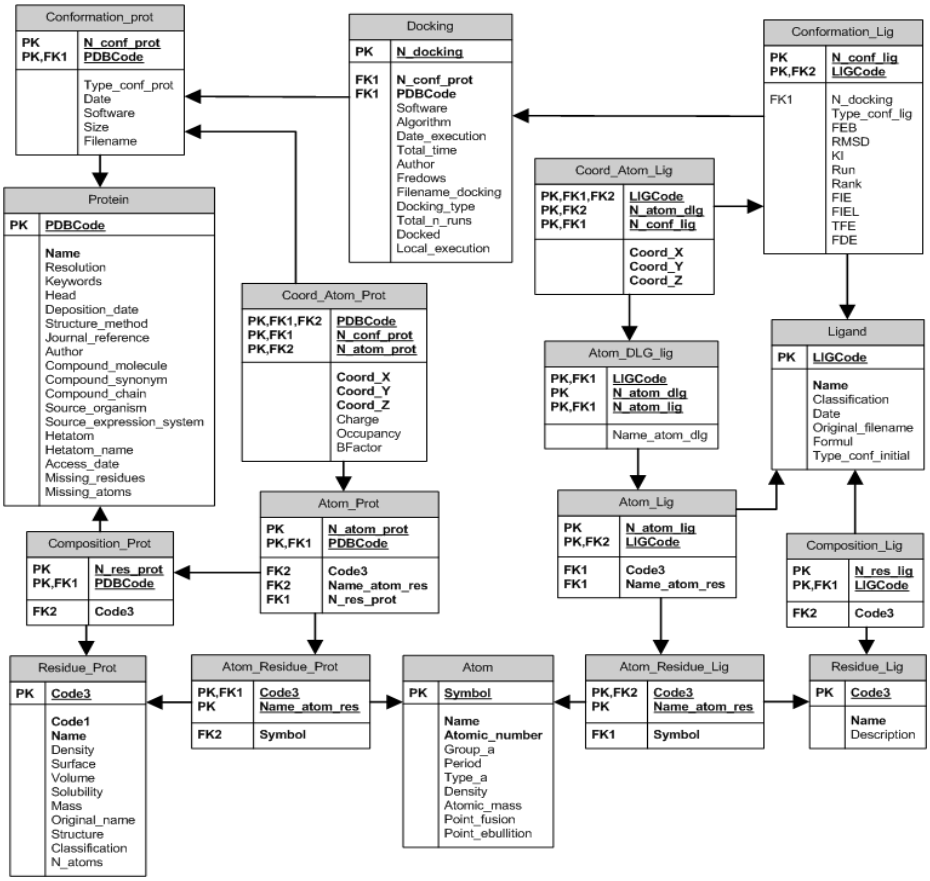


Figura 6.7. Modelo do banco de dados FReDD.

6.4.2. Experimentos com mineração de dados

A construção do repositório FReDD foi motivada para que os dados nele contidos pudessem ser facilmente acessados e preparados para serem utilizados por algoritmos de mineração de dados. O que se espera com a mineração é aumentar o entendimento a respeito do comportamento da flexibilidade do receptor sendo estudado e, com isso, contribuir para o aceleramento das execuções dos experimentos de docagem molecular. Para tanto, busca-se responder algumas perguntas, como:

- Como selecionar um subconjunto de conformações do receptor que seja o mais relevante para predizer se um dado ligante é promissor?
- Como, a partir de ligantes considerados promissores, selecionar outros que também tenham chance de serem promissores?

Para obter informações que direcionem para responder às perguntas acima, nesse trabalho são utilizadas regras de associação, árvores de decisão para classificação e árvore de decisão para regressão.

6.4.2.1. Pré-processamento de dados de docagem molecular

Os dados armazenados no FReDD precisam ser criteriosamente pré-processados para que os diferentes algoritmos de mineração aplicados possam induzir os melhores

modelos possíveis, de modo com que as perguntas enunciadas anteriormente tenham maior chance de serem respondidas.

O primeiro passo do pré-processamento é a escolha do atributo alvo, para as tarefas preditivas. Como sabe-se que uma das maneiras de avaliar os resultados de docagem é através dos valores de FEB, optou-se por utilizar esse atributo como alvo. Durante os 3.100 experimentos de docagem molecular do complexo InhA-NADH, considerando somente a melhor execução (*run*) de cada experimento, o valor médio da FEB foi de -12,9 ± 4,2 Kcal/mol, onde todos os experimentos de docagem convergiram para valores negativos de FEB, sendo o valor máximo de FEB (o mais negativo) de -20,6 Kcal/mol e ovalor mínimo de 0 Kcal/mol. Para árvores de regressão, considera-se o valor real do FEB (Machado et al, 2010b, Winck et al., 2010). Para árvores de decisão para classificação, esse atributo é discretizado em 5 classes, considerando moda e desvio padrão (Machado et al., 2010a) da distribuição da FEB. Para regras de associação o valor de FEB não é considerado.

A preparação dos atributos preditivos busca identificar as distâncias mínimas (representadas em angstroms, Å) entre os átomos de cada resíduo do receptor com os átomos do ligante (Machado et al, 2010b, Winck et al., 2010). Para a determinação dessa distância mínima são calculadas todas as distâncias entre todos os átomos do NADH e todos os átomos de cada resíduo, selecionando a menor entre elas. A Figura 6.8 ilustra esse conceito, apresentando as distâncias mínimas entre o ligante NADH e os resíduos Alanina 259 (ALA259) e Isoleucina 20 (ILE20) do receptor. Neste caso de exemplo, a menor distânicia entre o resíduo ILE20 e o NADH é de 2,35 Å e entre o resíduo ALA259 e o NADH é de 3,98 Å.

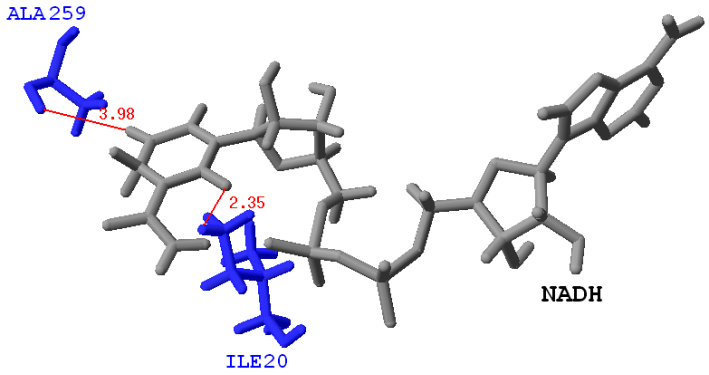


Figura 6.8. Distâncias atômicas entre o ligante NADH e o resíduo ILE20 e ALA259 do receptor InhA.

Para obter essas distâncias é necessário combinar todas as coordenadas do receptor com todas as coordenadas do ligante, a partir dos dados armazenados no repositório FReDD, conforme a consulta SQL ilustrada pela Figura 6.9. Por exemplo, se essa consulta é executada para o receptor InhA (PDBCode=1ENY) e o ligante NADH (LIGCode=NADH), tem-se uma combinação dos 12.424.800 registros de coordenadas do receptor (os 4.008 átomos multiplicados pelas 3.100 conformações), com os 161.200 registros de coordenadas do NADH (os 52 átomos do NADH multiplicados pelos 3100 resultados de docagem molecular para esse ligante). Isso significa que, ao realizar a

consulta ilustrada na Figura 6.9, será produzido mais de 2 trilhões de registros apenas para esse ligante. Os resultados obtidos precisam ser tratados de modo com que sejam transpostos e organizados em um total de 268 atributos (mais o atributo alvo), onde cada atributo corresponde a um resíduo do receptor, sendo que seus valores são atribuídos com as distâncias mínimas com relação ao ligante calculada para cada um desses resíduos. Cada instância corresponde a um experimento de docagem molecular.

```
select N_conf_prot, code3, N_res_prot, feb,
  min((car.Coord_x - cal.Coord_x)**2 +
      (car.Coord_y - cal.Coord_y)**2 +
      (car.Coord_z - cal.Coord_z)**2)
from Atom_Prot inner join Coord_Atom_Prot car
      using(PDBCode, N_atom_prot)
    inner join Conformation_prot
      using(PDBCode, N_conf_prot)
    inner join Docking
      using(PDBCode, N_conf_prot)
    inner join Conformation_lig using(N_docking)
    inner join Coord_Atom_Lig cal
      using(LIGCode, N_conf_lig)
where LIGCode = 'NADH'
  and PDBCode = '1ENY'
group by N_conf_prot, code3, N_res_prot, FEB;
```

Figura 6.9. Consulta SQL para obter as distâncias mínimas entre os átomos do ligante com os átimos do receptor.

A Tabela 6.1 ilustra parte do arquivo de entrada produzido. Esta tabela apresenta a estrutura básica, e inicial, do arquivo de entrada. Para cada experimento de mineração realizado, entretanto, o arquivo de entrada precisa ser ajustado conforme necessidades do algoritmo sendo utilizado.

Tabela 6.1. Exemplo do arquivo de entrada para o complexo InhA-NADH.

Exp. Docagem	...	ILE20	...	ALA259	...	FEB
1	...	2,35	...	3,98	...	-13,81
2	...	3,73	...	9,02	...	-9,75
...	...	...	...	...	...	...
3099	...	6.75	...	19,40	...	-7,59
3100	...	4,21	...	11,00	...	-14,56

6.4.2.2. Experimentos com regressão

Ao utilizar árvores de decisão sobre os dados de docagem molecular, busca-se descobrir quais são os resíduos do receptor e suas distâncias com relação ao ligante que mais contribuem para valores de FEB mais promissores. Considerando que os dados pré-processados contêm, essencialmente, valores numéricos, sobre os mesmos pode-se aplicar técnicas de regressão. Regressão é a tarefa de mineração mais utilizada para

predição de dados numéricos (Han e Kamber, 2006), sendo que existem diferentes algoritmos. Ávores modelo são algoritmos que utilizam-se da abordagem de árvores de decisão, sendo que nos nodos folha são apresentados modelos de regressão que correspondem a uma equação linear para o valor predito (Han e Kamber, 2006). Os modelos lineares (LM – *Linear Models*) podem ser utilizados para quantificar a contribuição de cada atributo para predizer o atributo alvo, no caso o FEB.

Neste trabalho é utilizado o algoritmo M5P (Quinlan, 1992), implementação de árvores modelo no WEKA. Dentre os parâmetros disponíveis para este algoritmo, concentramos na calibragem dos parâmetros relacionados à legibilidade e precisão das árvores modelo induzidas. Portanto, definimos o número mínimo de instâncias para 100. Este tamanho está relacionado com o tamanho da árvore modelo resultante e o número de LM produzidos.

Para reduzir os atributos preditivos, aplicou-se uma seleção de atributos baseada no contexto (Winck et al, 2010), considerando as distâncias mínimas dos resíduos. O maior valor de distância que permite um contato biologicamente significante entre os átomos do receptor e do ligante é 4Å (da Silveira et al., 2009). Assim, os dados de entrada gerados no pré-processamento inicial (Tabela 6.1) são refinados, eliminando todos aqueles atributos cuja distância mínima seja maior do que 4Å.

Executando o M5P sobre esses dados, o modelo induzido produziu uma árvore com 10 nodos e 11 LM, e com uma correlação de 92%. Para melhor entender a saída do M5P, a Figura 6.10 ilustra a árvore modelo induzida para o ligante NADH e a equação 1 ilustra o LM1 gerado. Cada nó interno da árvore (Figura 6.10) corresponde ao código de três letras do aminoácido e seu número na seqüência receptor. Cada ramo representa o limite de distância do ligante para o receptor.

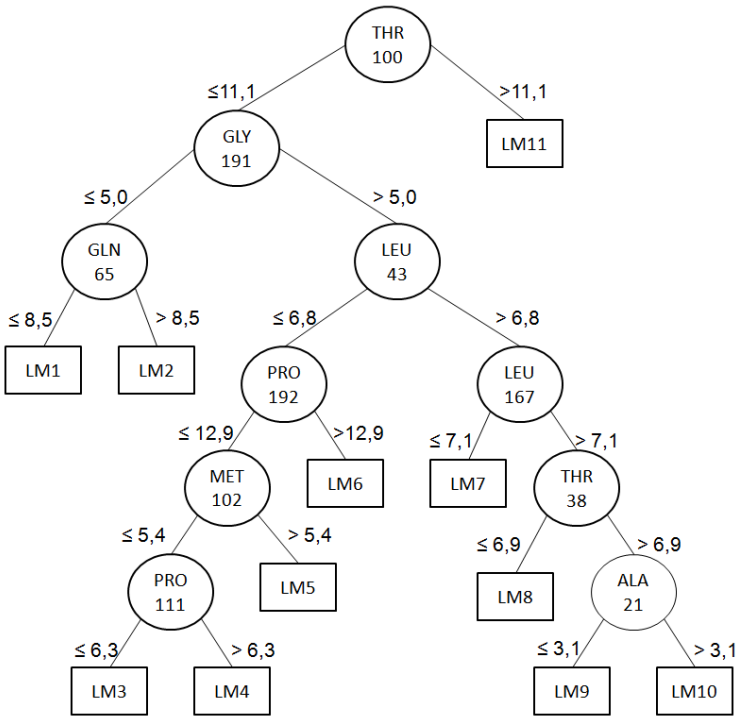


Figura 6.10. Árvore de regressão do complexo InhA-NADH.

```
LM num: 1
FEB = 0.0019 * ILE14 + 0.0141 * SER19 - 0.0129 * ALA21 +
      0.0034 * PHE22 + 0.0015 * HIE23 + 0.0085 * ILE24 +
      0.0157 * THR38 + 0.0165 * LEU43 + 0.0081 * LEU45 +
      0.0027 * LEU62 + 0.0844 * GLN65 - 0.0039 * PHE96 -
      0.0035 * MET97 0.0233 * PRO98 + 0.0179 * THR100 -
      0.0051 * GLY101 - 0.0138 * MET102 + 0.0155 * PRO111 -
      0.0016 * ASP114 - 0.0014 * LYS117 + 0.0125 * GLY118 -
      0.0012 * MET146 - 0.0019 * ALA163 + 0.0855 * LYS164 +
      0.0386 * LEU167 + 0.0338 * ALA190 - 0.0116 * GLY191 +
      0.0358 * PRO192 + 0.0014 * ET198 - 0.0025 * ALA200 +
      0.0142 * ILE201 + 0.0037 * ALA205 - 0.0012 * VAL237 - 13.6659
```
(1)

Pra analisar as árvores e identificar quais LM que indicam valores de FEB mais promissores, é efetuado um pós-processamento sobre os modelos induzidos (Machado et al., 2010; Machado et al., 2010b), com base no valor médio de FEB. Para o caso da Figura 6.10, o modelo linear mais representativo é o LM 11. Nesse sentido, a partir da Figura 6.10 pode-se observar, por exemplo, que sempre que o resíduo Treonina 100 (THR100) está a uma distância superior a 11Å do ligante NADH o modelo linear correspondente é o LM11. Isso significa que um resíduo do receptor que, aparentemente, não é importante (já que está distante do ligante), passa a ser crucial para apontar quais são as conformações que levam a FEB mais promissoras.

6.4.2.3. Experimentos com classificação

Ao submeter o conjunto de dados a um algoritmo de árvore de decisão, temos o mesmo objetivo da regressão: identificar quais conformações podem ter melhores resultados de FEB em experimentos de docagem moleculares. Para efetuar esses experimentos utilizamos o algoritmo J48, implementação do C4.5 (Quinlan, 1986) no WEKA.

Como classificação utiliza-se de classes categóricas, o conjunto de dados pré-processados precisa ser ajustado à classificação. Nesse sentido, o valor de FEB é discretizado em 5 categorias, considerando-se moda e desvio padrão (Machado et al, 2010a). Essas categorias dividem-se em:

- excelente (E);
- bom (B);
- regular (Re);
- ruim (R); e
- muito ruim (MR).

Para gerar árvores mais legíveis, configuramos o parâmetro relacionado ao número mínimo de objetos para 50, mantendo os demais parâmetros em sua configuração original. O algoritmo foi executado considerando a opção de validação cruzada com 10 partes. Como resultado para o NADH obteve-se árvores com acurácia de 73%. A Figura 6.11 ilustra a árvore de decisão induzida para o complexo InhA-NADH. Por essa árvore observa-se que, assim como na árvore modelo produzida (Figura 6.10), o nodo raiz também é formado pelo resíduo THR100 e, embora ele seja um resíduo distante do ligante, é importante para predizer bons resultados de FEB.

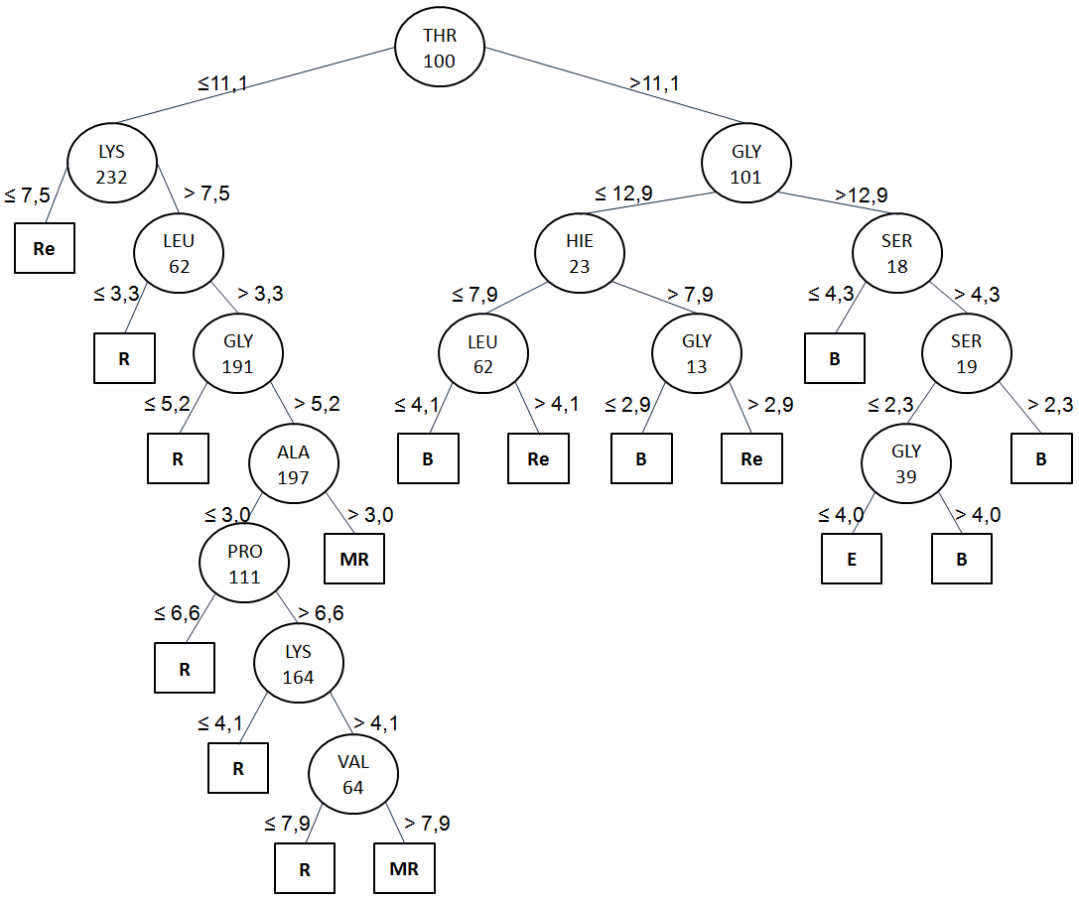


Figura 6.11. Árvore de regressão para o complexo InhA-NADH.

6.4.2.4. Experimentos com regras de associação

Ao utilizar regras de associação sobre o conjunto de dados apresentado nesse trabalho, busca-se identificar relações de interação entre resíduos. Para essas regras, o conjunto de dados utilizado foi preparado em duas etapas. Primeiramente o atributo alvo (FEB) foi removido. Em seguida os valores dos atributos passaram a conter valores binários, indicando a existência de interação dos resíduos do receptor com o ligante: 0 para distâncias maiores que 4Å; e 1 para distâncias menores ou igual a 4Å (Machado et al., 2008).

Os arquivos preparados foram submetidos ao algoritmo *Apriori* (Agrawal, 1993) do WEKA, ajustando o valor de suporte para 0,005 e confiança para 0,9 e um número máximo de 1000 regras. As regras produzidas foram pós-processadas, seguindo a abordagem proposta em Winck et al. (2010a), onde as regras mais especializadas são removidas, atingindo modelos mais enxutos e eficazes. Pra exemplificar algumas regras geradas para o ligante NADH, ilustramos três regras representativas:

- THR100=0 → ILE94=1;
- THR100=0 → SER19=1;
- THR100=0→THR195=1;

Por essas regras nota-se que, quando o resíduo THR100 não interage com o NADH, os resíduos ILE94, SER19 e THR195 interagem. Isso ratifica os padrões identificados através das árvores de que o resíduo THR100, mesmo não interagindo com o ligante, é representativo para indicar aqueles resíduos que interagem.

Muitas outras regras podem ser extraídas. Embora essas regras não estabeleçam relações entre os resíduos do receptor e os valores de FEB, elas podem ser úteis para indicar quais são os resíduos que mais interagem com o ligante sendo estudado. Essa informação pode ser útil na busca de novos ligantes para este receptor, como no trabalho de Quevedo et al. (2010). Em um trabalho anterior (Machado et al, 2008), ao utilizar apenas 40 resultados de experimentos de docagem e 40 conformações do receptor, foi possível extrair conhecimento sobre os resíduos de receptor que mais interagem com os ligantes estudados.

6.5. Considerações

Neste trabalho foram descritas todas as etapas envolvidas em um processo de KDD aplicado à bioinformática no contexto de desenho racional de fármacos, utilizando especificamente resultados de docagem molecular. Os experimentos de docagem molecular com um receptor flexível com 3.100 conformações e um ligante, o complexo InhA-NADH, foram executados previamente e os resultados todos armazenados um repositório apropriado. Os dados armazenados no repositório FReDD foram então preprocessados e assim preparados para serem utilizados nos algoritmos de mineração. Os algoritmos de mineração utilizados foram: M5P de árvores de regressão, J48 para árvores de decisão e Apriori para regras de associação.

Os resultados obtidos com as diferentes técnicas de mineração aplicadas mostram alguns exemplos de informações que podem ser obtidas sobre os experimentos de docagem molecular, que não seria possível de serem extraídas sem um processo de KDD. Um exemplo são os resíduos que aparecem tanto na árvore de regressão quanto na árvore de decisão do NADH, que são resíduos que em uma inspeção visual com uma conformação desse receptor e o NADH não parecem estar em contato com o mesmo (não estão a uma distância menor do que 4Å do ligante). Com as informações obtidas durante esse processo de KDD espera-se que, no futuro, seja possível acelerar os experimentos de docagem molecular, utilizando com novos e diferentes ligantes as conformações indicadas como mais promissoras nos experimentos já executados.

6.6. Biografias

Ana Trindade Winck possui graduação em Ciência da Computação pela Universidade Feevale (2005) e mestrado em Ciência da Computação pela PUCRS (2007). Atualmente é aluna de doutorado em Ciência da Computação pela PUCRS, com início em 2008 e previsão de conclusão em 2011. É integrante do GPIN, e desenvolve sua tese com ênfase no pré-processamento de dados biológicos. Tem desenvolvido pesquisas em catalogação e mineração de dados de docagem molecular. Possui interesse em bioinformática e mineração de dados. Tem proferiu palestras e ministrou minicurso nestas áreas.

Karina dos Santos Machado possui graduação em Engenharia de Computação pela FURG (2004) e mestrado em Ciência da Computação pela PUCRS (2007). Atualmente é aluna do último ano de doutorado do curso de Ciência da Computação da PUCRS,

integrante do LABIO. Desenvolve sua tese na aplicação de diferentes técnicas de mineração de dados à seleção de conformações do receptor. Tem desenvolvido pesquisas em catalogação e mineração de dados de docagem molecular. Possui interesse em bioinformática e mineração de dados, e experiência em ministrar minicurso na área.

Duncan Dubugras Alcoba Ruiz possui graduação em Engenharia Elétrica pela Universidade Federal do Rio Grande do Sul (1983), mestrado em Ciências da Computação pela Universidade Federal do Rio Grande do Sul (1987) e doutorado em Ciências da Computação pela UFRGS (1995). Atualmente é professor adjunto da PUCRS, onde coordena o GPIN. Tem experiência na área de Ciência da Computação, com ênfase em Banco de Dados, atuando principalmente nos seguintes temas: banco de dados, *workflow modeling*, bancos de dados temporais, engenharia de software e KDD.

Osmar Norberto de Souza possui graduação em Física pela Universidade Federal de Minas Gerais (1987), especialização em Biotecnologia Moderna pelo Centro de Biotecnologia da UFRGS (1988), mestrado e doutorado em Biofísica Molecular Computacional - University of London (1994). Trabalhou de 1995 a 1998, no Environmental Molecular Sciences Laboratory do Pacific Northwest National Laboratory do Departamento de Energia (DOE) dos EUA e de 1999 a 2001, no Laboratório Nacional de Luz Síncrotron (LNLS), Campinas, SP. Desde 2002 é professor adjunto da PUCRS, onde coordena o LABIO. Atualmente é bolsista de produtividade em pesquisa CNpQ - Nível 2. Tem experiência nas áreas de Bioinformática Estrutural e Biofísica, com ênfase em Biofísica Molecular Computacional, atuando principalmente nos seguintes temas: simulação computacional por dinâmica molecular, estrutura e dinâmica de proteínas e complexos proteína-ligante, protocolos e algoritmos computacionais para a predição da estrutura tridimensional de proteínas, planejamento de fármacos assistido por computador, tuberculose e malária.

Referências

Agrawal, R., Imielinski, T., Swami, A. (1993). "Mining association rules between sets of items in large databases". In Conference on Management of Data, Washington DC, 207-216.

Alpaydin, E. Introduction to machine learning. The Mit Press, Cambridge, 2004.

Amaro, R.E., Baron, R., McCammon J.A. (2008). "An Improved Relaxed Complex Scheme for Receptor Flexibility in Computer-aided Drug Design." Journal of Computer Aided Molecular Design, 22:693-705.

Balakin, K. V. Pharmaceutical data mining: approaches and applications for drug discovery. John Wiley & Sons, New York, 2009.

Benson, D.A., Karsch-Mizrachi, I., Lipman D.J., Ostell J., Wheeler D.L. (2008). "GenBank". Nucleic Acids Res., 36, Database issue D25-D30.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., Bourne, P. E. (2000). "PDB - Protein Data Bank". Nucleic Acids Research, 28:235-242.

Broughton, H. B. (2000). "A Method for Including Protein Flexibility in Protein-ligand Docking: Improving Tools for Database Mining and Virtual Screening". Journal of Molecular Graphics and Modelling, 18: 247-257.

Chandrika, B., Subramanian, J., Sharma, S.D. (2009). “Managing Protein Flexibility in Docking and its Applications” Drug Discovery Today, 14:394-400.

da Silveira, C.H., Pires, D.E.V., Minardi, R.C., Ribeiro, C., Veloso, C.J.M., Lopes, J.C.D., Meira Jr, W., Neshich, G., Ramos, C.H.I., Habesch, R., Santoro, M.M. (2009). “Protein Cutoff Scanning: A comparative Analysis of Cutoff Dependent and Cutoff Free Methods for Prospecting Contacts in Proteins”. Proteins, 74:727-743.

Dessen, A., Quemard, A., Blanchar, J.S. Jacobs Jr, W.R., Sacchenttini, J.C. (1995). “Crystal structure and function of the isoniazid target of Mycobacterium tuberculosis”. Science. 267(5204):1638-41

Drews, J. (2003). “Strategic trends in the drug industry”. Drug Discovery Today, 8:411-420.

Fayyad, U., Piatestsky-S, G., Smyth, P. (1996). “The KDD process for extracting useful knowledge from volumes of data”. Communications of the ACM, 39:27-34.

Freitas, A., Wieser, D., Apweiler, R. (2010). “On the importance of comprehensible classification models for protein function prediction”. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 7:172-182.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutmann, P., Witten, I. (2009). “The WEKA data mining software: an update”. SIGKDD Explorations, 11:10-18.

Han, J., Kamber, M. Data Mining: concepts and techniques. Morgan & Kaufmann, San Francisco, 2006.

Hunter, L. Artificial intelligence and molecular biology. American Association for Artificial Intelligence. Menlo Park, CA, USA, 1993.

Inmon, W.H. Como construir um data warehouse. Campus, Rio de Janeiro, 1997.

Kimball, R., Ross, M. The datawarehouse toolkit: the complete guide to dimensional modeling. John Wiley & Sons, New York, 2002.

Kuntz, I. D. (1992). “Structure-based strategies for drug design and discovery”. Science, 257: 1078-1082.

Kuo, M.R., Morbidoni, H. R., Alland, D., Sneddon,S. F., Gourlie, B. B., Staveski, M. M., Leonard, M., Gregory,J. S., Janjigian,A. D., Yee, C., Musser, J.M., Kreiswirth, B., Iwamoto, B., Perozzo, R., Jacobs Jr, W.R., Sacchettini, J.C., Fodock, D.A. (2003). “Targeting Tuberculosis and Malaria through Inhibition of Enoyl Reductase: Compound Activity and Structural Data”. Journal of Biological Chemistry, 278: 20851-20859.

Lesk, A. Introduction to bioinformatics. Oxford University Press, 2002.

Lin, J-H., Perryman, A.L., Schames, J.R., McCammon, J.A. (2002) “Computational drug design accommodating receptor flexibility: the relaxed complex scheme” Journal of American Chemical Society. 124:5632-5633.

Luscombe, N.M., Greenbaum, D., Gerstein, M. (2001). “What is bioinformatics? A proposed definition and overview of the field” Methods Information in Medicine, 40:346-358.

Lybrand, T.P. (1995). “Ligand-protein docking and rational drug design”. Current Opinion in Structural Biology, 5: 224-228.

Machado, K.S., Schroeder, E.K., Ruiz, D.D., Norberto de Souza, O. (2007). “Automating molecular docking with explicit receptor flexibility using scientific workflows”. In: Second Brazilian Symposium on Bioinformatics, LNCS, 1-11, Rio de Janeiro.

Machado, K.S., Schroeder, E.K., Ruiz, D.D., Winck, A.T., Norberto de Souza, O. (2008). “Extracting information from flexible ligand-flexible receptor docking experiments”. In: Third Brazilian Symposium on Bioinformatics, LNCS, 104-112, Santo André.

Machado, K.S., Winck, A.T., Ruiz, D.D., Norberto de Souza, O. (2010a). “Discretization of Flexible-Receptor Docking Data”. In: Fifth Brazilian Symposium on Bioinformatics, LNCS, 6268:75-79.

Machado, K.S., Winck, A.T., Ruiz, D.D., Norberto de Souza, O. (2010b). “Mining flexible-receptor docking experiments to select promising protein receptor snapshots”. BMC Genomics. To be published.

Machado, K.S., Winck, A.T., Ruiz, D.D., Norberto de Souza, O. (2010). “Applying model trees on flexible-receptor docking experiments to select promising protein receptor snapshots” In: ISCB Latin America, 66-66, Montevideo.

Morris, G.M., Goodsell, D.S., Halliday, R.S., Huey, R., Hart, W.E., Belew, R.K., Olson, A.J. (1998). “Automated Docking Using a Lamarckian Genetic Algorithm and Empirical Binding Free Energy Function”. Journal of Computational Chemistry, 19: 1639-1662.

Mount, DW. Bioinformatics: Sequence and Genome Analysis. Cold Spring Harbor Laboratory Press. New York. 2004.

Oliveira, J.S., Moreira, I.S., Santos, D.S., Basso, L.A. (2007). “Enoyl reductases as targets for the development of anti-tubercular and anti-malarial agents”. Current Drug Targets, 8:399-411.

Oliveira, J.S., Souza, E.H.S., Basso, L.A., Palaci, M., Dietze, R., Santos, D.S., Moreira, I. (2004). “An Inorganic Iron Complex that Inhibits Wild-type and an Isoniazid-resistant Mutant 2-transenoyl-ACP (CoA) Reductase from Mycobacterium tuberculosis”. Chemical Communication, 15:312-313.

Pearlman, D.A., Case, D.A, Caldwell, J.W., Ross, W.R., Cheatham, T.E.., DeBolt, S., Fergunson, D., Seibel, G., Kollman, P. (1995). “AMBER, a computer program for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to elucidate the structures and energies of molecules”. Computer Physics Communications., 91:1-41.

Quevedo C.V., Winck, A.T., Machado, K.S., Norberto de Souza, O., Ruiz D.D. (2010). “A study of molecular descriptor to rank candidate ligands to inhibit the InhA Receptor”. In: ISCB Latin America, 79-79, Montevideo.

Quinlan, J.R. (1986). “Introduction to decision trees”. Machine Learning, 1:81-106.

Quinlan, J.R. (1992). “Learning with continuous classes” In Fifth Australian Joint Conference on Aritifical Intelligence, Singapure, 343-348.

Schroeder, E.K., Basso, L.A., Santos, D.S., Norberto de Souza, O. (2005). "Molecular Dynamics Simulation Studies of the Wild-Type, I21V, and I16T Mutants of Isoniazid-Resistant Mycobacterium tuberculosis Enoyl Reductase (InhA) in Complex with NADH: Toward the Understanding of NADH-InhA Different Affinities". Biophysics Journal, 89:876-884.

Tan, P-N., Steinbach, M., Kumar, V. Introduction to data mining. Addison-wesley, Boston, 2006.

Totrov, M., Abagyan, R. (2008). "Flexible Ligand Docking to Multiple Receptor Conformations: a Practical Alternative". Current Opinion on Structural Biology, 18:178-184.

van Gunsteren, W.F., Berendsen, H.J.C. (1990). "Computer Simulation of Molecular Dynamics: Methodology, Applications and Perspectives in Chemistry". Angewandt Chemie International Edition, 29:992-1023.

Wang, F., Langley, R., Gulten, G., Dover, L.G., Besra, G.S., Jacobs Jr, W.R., Sacchettini, J.C. (2007). "Mechanism of thioamide drug action against tuberculosis and leprosy". Journal of Experimental Medicine, 204:73-78.

Wang, J. T. L, Zaki, M. J. and Toivonen, H. T. T. Data mining in bioinformatics: advances information and knowledge processing, Springer, 2005.

Winck, A.T., Machado, K.S., Norberto de Souza, O., Ruiz, D.D. (2010). "A context-based preprocessing on flexible-receptor docking data". In: ISCB Latin America, 68-68, Montevideo.

Winck, A.T., Machado, K.S., Norberto de Souza, O., Ruiz, D.D. (2009). "FReDD: supporting mining strategies through a flexible-receptor docking database". In: Fourth Brazilian Symposium on Bioinformatics, LNCS, 6098:143-146, Porto Alegre.

Winck, A.T., Machado, K.S., Ruiz, D.D., Strube de Lima, V.L. (2010a). "Association rules to identify receptor and ligand structures through named entities recognition" In: The Twenty Third International Conference on Industrial Engineering & Other Applications of Applied Intelligent Systems, LNCS, 119-128, Córdoba.

Chapter 7

An Introduction to Model-driven Development of User Interfaces to realize Multimodal and Context-sensitive Applications for Smart Environments

Sebastian Feuerstack

Abstract

Computer systems as we know them are currently changing from single systems running a set of applications to complex networks of connected devices heavily influencing the users' everyday lifes. This leads to new requirements for applications and human-computer interaction. Networks of heterogeneous devices in homes and offices require flexible applications that can adapt to changing environments supporting the user in his daily life - anywhere, on any device and at anytime. This requires interactive systems to be able to switch modalities (between tangible, speech, and multi-touch interaction modes) and devices on demand of the user and to enable user interface distribution combining several devices to control one interactive system.

The tutorial gives an introduction to the model-driven development of user interfaces (MDDUI). The tutorial consists of two parts: First, beneath introducing the basic terms and definitions, multi-targeting processes and tools for the user-centered design of interfaces for different context of use are presented. By a discussion about the pros and cons of multi-targeting the second part about the actual switch from design-time to run-time-based approaches in MDDUI is introduced and two exemplary approaches are presented. Finally, current challenges in MDDUI are discussed to motivate the participation in this promising research field.

7.1. Introduction

The change from single computing systems towards smart environments connecting complex networks of devices leads to new requirements for applications and human-computer interaction. A smart environment extends the potential interaction options for

the user from a small amount of isolated utilized devices to combinations of devices that enable addressing a mixture of modalities, like speech, gesture or graphical-driven interaction.

Together with various interconnected sensors that make the environment aware of the user, interactions can be embedded in the environment: Not a specific device, but the whole environment can be used as the interface. This enables a user to seamlessly interact with a computer by choosing and combining a heterogeneous set of interaction devices and modalities based on his preferences.

Addressing various different devices and modalities thus requires the possibility to adapt the user interface to support the specific features of the device's software platform and the provided interaction capabilities. Model-driven user interface development (MDDUI), practiced for a long time in Human-Computer Interaction seem to be a promising approach to support software developers, developing interactive applications. It offers a declarative way of modelling such multi-platform user-interfaces and aims at modelling the different aspects of the user interface on certain levels of abstraction.

Abstraction is a very natural way that people usually like to do to manage too complex entities. To manage complex problems people usually start to discover the main aspects that should be taken into account as well as their relations. An example for identifying models in our daily practice is for instance our every day planning. After waking up we typically start the new day by thinking about the main activities that we would like to perform [53].

With the help of a software engineering process that changes the focus from (manual) implementation to the tool-driven design of models that can be directly executed by a run-time-system, software developers can be supported to address these challenges. This discipline of Engineering for Human-Computer Interaction (EHCI) is a crossroad of two disciplines: Human Computer Interaction (HCI) and Software Engineering (SE).

This article is published to accompany a tutorial held at the Webmedia 2010 conference to introduce the concepts of MDDUI to a broader audience. Therefore I start by reviewing some previous work in this area to introduce the basic terms and concepts of MDDUI. Since I cannot cite all relevant work for the sake of brevity I focus on presenting the initial efforts of MDDUI to generate multiple user interfaces for different devices to reflect the advantages and disadvantages of MDDUI in general and in relation to be applicable for generating and running context-sensitive and multimodal interfaces in smart environments. This discussion is used to motivate the recent shift of interest in MDDUI run-time systems that awake the declarative design models alive to support interface manipulation and a more flexible reaction on context-changes at run-time that I devote special attention to. I refer to my own research results as well as of other research groups to explain the new possibilities multimodal and context-sensitive interaction in smart environments that can be tackled by MDDUI run-time systems. Finally I end up with discussing three basic research challenges that need to be addressed in the next years to enable a seamless interaction in smart environments.

7.2. Model-driven User Interface Development (MDDUI)

The development of interactive systems is a complex problem that is continuously growing with the evolving technology enabling to consider the context of use during interaction or multi-modal access to an application. Model-based approaches, practiced since the late 1980's in Human-Computer Interaction seem to be a promising approach to support software developers, developing interactive applications.

MDDUI is driven by two basic ideas: First, by shifting the perspective in user interface construction from a developer to a user-centric one and second, by reducing the development effort for the creation of user interface that quickly gets cumbersome if an interface should run ad on more than one context (for instance on different devices).

User interface builders that enable a developer to create interfaces and to organize widgets in several dialog boxes are the general approach for a developer-centric approach. The organization of the user interface layouts mainly relies on the designer's experience or on external and therefore loosely coupled design documentation. Different to this, user-centered design environments focus to answer how user interface elements in a certain dialog box can be used to accomplish a particular user task [56].

A design process that is concerned with supporting the creation of interfaces for several contexts has been defined as muti-targeting. Multi-targeting describes a coordinated process of building user interfaces for a set of given context of use. A context of use is defined as a triple (U, P, E) where U represents any user stereotype, P, any computing platform, and E, the physical environment in which the user is carrying out her task with the designated platform [44].

The basic concept of addressing both, the user-centric development perspective and multi-targeting is to model the different aspects of the user interface on certain levels of abstraction. Each level of abstraction is described by a model that offers a declarative way of designing an interface. Therefore research in MDDUI is in general concerned with two basic efforts:

1. The identification of suitable sets of model abstractions and their relations for analyzing, designing and evaluating interactive systems
2. The specification of software engineering processes that change the focus from (manual) implementation to the tool-driven design of models

In the last 20 years of research various model-based approaches have been proposed. An overview paper has been written by [67]. In 2003 the Cameleon Reference Framework has been proposed [13] that serves as a reference for classifying MDDUI approaches that support multi-targeting or multiple contexts of use and has been applied to several MDDUI approaches at this time.

7.2.1 The Cameleon Unifying Reference Framework

The Cameleon Reference Framework [13] defines several design steps for the development of multi-context interactive applications. The basic development process is divided into four steps and is illustrated for two contexts of use in figure 1.

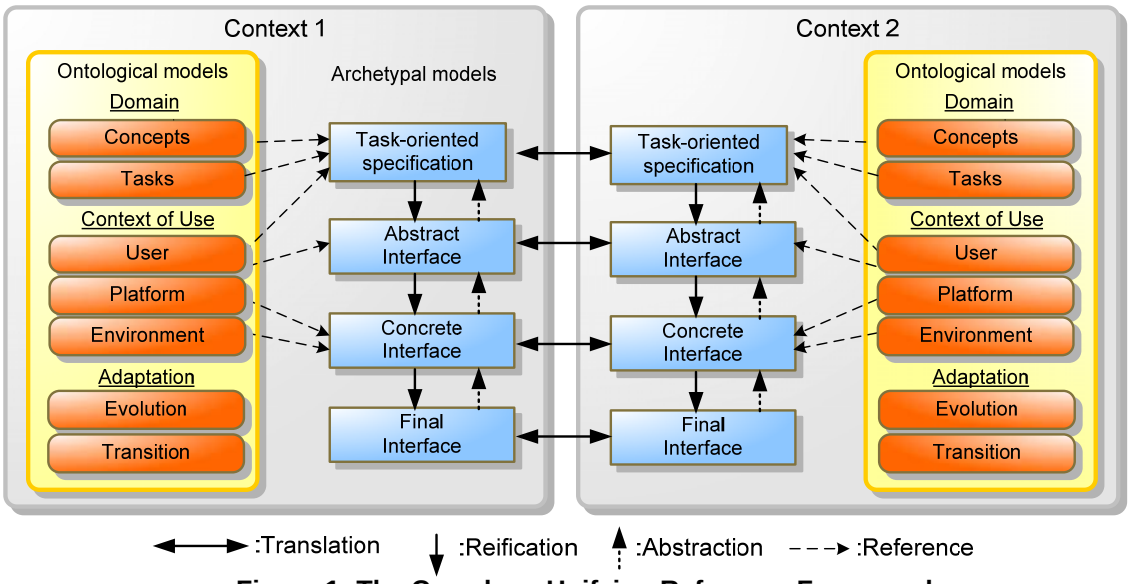


Figure 1: The Cameleon Unifying Reference Framework

7.2.2 Common declarative models of MDDUI approaches

The Framework describes every development step as a manipulation of the artifacts of interests in form of a model or a user interface representation. In distinguishes between ontological and archetypal models. The former one are meta-models and independent of any domain and interactive system, the latter are instantiated ontological models and are dependent on an interactive system for a given domain. Like illustrated in figure 1 there are three basic ontological models: Domain, Context, and Adaptation.

The *domain model* describes concepts and user tasks relative to a domain. A concept might be for instance an UML class diagram, an entity-relationship, or an ontology. The *task model* might be a model that breaks down a user's goals into several activities that need to be either performed by him or by an interactive system and organizes each goal through tasks into logical activities. An instantiated task model might describe how a concrete goal – for instance: "organize meeting" has to be performed by several subsequent tasks in a certain order. Each task is performed either by an interactive system or a user that might be described by a role specification. Tasks are associated to objects instantiated from a concept model and get manipulated during a task is performing.

The ontological *context of use models* enable reasoning about the *users, platforms, and environments* and define the targets of an MDDUI process. The user is relevant for the task specification (e.g. by a role specification) as described earlier. The platform model describes the targeted hardware (device capabilities like screen resolution) and software platform (like e.g. the operating system). Further on, elementary platforms (interaction devices) can be composed to clusters, to be addressed as a single target. Part of the software platform is the *interactor model* that describes the interactors available for the presentation of the user interface. The archetypal realization of an interactor model is for instance a toolkit (e.g. Java Swing or HTML), which contains several widgets (a list box, a button, a voice menu, or a set of navigation gestures). The description of an interactor includes it's look-and-feel, it's input and output capabilities like e.g. required space, the required modes (mouse, keyboard, speech), and side-effects to the context of

use (e.g. additional noise level in case of a speech output). Beneath information about the user, the platform and the environment of the original context of use model definition by [13] the ongoing time is another relevant context-information. It influences the relevance of information that is transmitted by an interface to the user and can be used to adapt the interface by removing information that got irrelevant or highlighting highly relevant parts for a particular time span.

Finally the environment model considers the physical surroundings of the user and the platforms available. This includes for instance the distances between platforms and the users or environmental conditions like noise or lighting level.

The Cameleon Framework further defines two ontological adaptation models. The first one, the evolution model defines triggers based on context of use changes and appropriate reactions that specify a configuration of the interactive system to switch to. The second, the transition model ensures by prologues and epilogues for each context-transition how to handle discontinuities that might occur at a context-change.

An interactive application design as it is described by the Cameleon Framework is based on *initial models* and *transient models*. Initial models a are based on manually entered design information whereas transient models are the result of a model-to-model transformation. There are four *design models* that are often used in MDDUI:

- A Tasks and Concepts model that combines tasks that have to be carried out by the final interactive system with a description of the domain-oriented concepts that are needed by these tasks. Concepts are represented as classes that get instantiated to objects used to reflect the concept manipulation.
- An Abstract User Interface (AUI) is used to describe the logical structure without targeting it to a specific modality or even a platform. An AUI refers to interaction objects that are described in terms of abstract interactors. The AUI groups the subtasks of the task model into interaction spaces (presentation units) according to a set of criteria like task model structural patterns, cognitive work load analysis or identification of the semantic relationships between tasks. Further on the AUI defines a navigation scheme between the interaction spaces and selects Abstract Interaction Objects (AIO) for each of the concepts that have to be independent of any mode of interaction.
- The Concrete User Interface (CUI) that replaces each abstract interactor with a concrete interaction object that is dependent of a modality and a set of associated platforms that implement the same modality and are similar regarding their capabilities (for instance the screen size, and the control features). The Concrete User Interface model (CUI) concretizes the AUI and defines the widget layout and interface navigation through these widgets. Different to the AUI, the CUI is designed for a concrete modality but independent of a specific platform. The CUI is used to present the final look and feel of the user interface but has to be considered as a mockup that only runs within a particular environment.
- On the lowest level of abstraction the Final User Interface (FUI) represents the operational user interface that can be run on a platform either by interpretation

through a renderer (for instance a web-browser) or by execution (after compilation into binary code).

7.2.3 Multi-path development processes

In the last section I have introduced a lot of design models that are relevant in MDDUI to realize an interactive application. But what methods exist to generate or manually design these models? There have been proposed several techniques that can be used that can be used selective – based on requirements ore preconditions of a project or be can be combined and applied in one process: Transformations, Translations, Graceful Degration, and Factorization. Further on different Entry Points mark models that can be used as an initial model to start the process. Finally by referring to examples the progress in the MDDUI process can be made more transparent and feedback can be considered earlier.

Reification, Abstraction, and Translation

There are three different types of model-to-model transformations that are applied to generate or manipulate a design model based on information of another source model: Reification, Abstraction and Translation.

Reification is done by performing forward engineering. Thus, like depicted in figure 1 the designer starts with an abstract design model and transforms it - after it has been specified completely – to a more concrete design model. Concrete models in MDDUI are more shaped to a particular platform and therefore contain more specific details as its abstract successors.

Contrary to reification is *abstraction*, which is in terms of software engineering usually called reverse engineering. Abstraction is a process of generalization of concrete artifacts. It is often an expensive one in terms of effort and complexity. This is because usually the abstraction process starts with an existing system and with the goal to obtain a conceptual description of its design for analysis and evaluation purposes. There are tools available to support model abstraction like ReversiXML [10] and WebRevenge [38] that support partial automation for the abstraction of applications for the web, but with the increasingly usage of dynamic pages driven by technologies - such as AJAX – show the limits of automated reverse engineering approaches. A different approach for reverse-engineering that relies on screenshots and manual annotation to support an automated usability analysis based on abstracted models has been implemented in the MeMo-workbench [29].

Translation is used to transform a design model that is designed for a specific context of use to a different one without switching the level of abstraction. An example would be an interface migration between two targets - for instance between a web page and a speech interface. Although a translation maintains the level of abstraction it can be performed for instance on the task and concepts level to change the tasks that are relevant for each target (which allows maximal flexibility). Or a translation is executed on a more concrete level like for instance to change widgets representing a list from a pull-down to a radio group between different HTML targets. The latter one enables much less flexibility since the application tasks remain the same - but offers more consistency to the user.

Multi-path development offers a lot of possibilities just by model-to-model transformations. Beneath the straight forward- and reverse-engineering approaches, in practice often a mixture between these methods is performed. This is because of changing requirements during the development (such as considering more targets). Additionally the process of transformation has been adjusted to especially to overcome the complexity that is introduced by the high level of (model-based) abstractions as well as the relatively long development process through various models. In the following several methods for reducing the complexity are presented: Graceful Degradation, Factorization, Entry Points and Development by Examples.

Graceful Degradation

The method of Graceful Degradation addresses the trade-off between continuity and adaptation and has been introduced first for safety-critical systems like for instance systems in an aerospace. There Graceful Degradation describes the ability of a system to continue its services proportionally to its components that fail or are malfunctioning. Thus, under bad conditions the system is not expected to fail completely but continue providing at least its basic functions that are offered by all the non-defect components.

Graceful Degradation can be applied to user interfaces as well [16]. There it consists of specifying the user interface for the least constrained platform first and than it requires to apply transformation rules to the original interface to produce interfaces to the more constrained platforms. These transformation rules include: Splitting rules, interactor and image transformation rules, moving and resizing rules to reuse the user interface, and removal rules [31].

Factorization and Decoration

The idea of Factorization is to support multi-targeting from a “specific-target-at-a-time” approach [69]. This enables the designers to focus on one target after another. After the second target design model(s) have been realized (independently from the first), common parts are identified by comparing the design models of both targets and are factored out in a separate model. Instead of factoring out common parts to a separate model, by Decoration the designers identify a reference target, which is usually like in Graceful Degradation the least constrained target, and express exceptions by annotations to the reference model. Factorization is similar to a refactoring process, where repeating parts are cleaned up and placed in a separate component that is referenced by each component from that the code has been factored out. A detailed discussion about factorization for different context of use for task modeling has been presented by [63]. An example of decorating task models by annotations to consider different context of use has been proposed by [18].

Entry Points

An Entry Point defines an initial design model that is used to start a MDDUI process. The Cameleon Framework allows entry points at any level of abstraction and in principle engineering from any entry point in all directions (abstraction, reification, translation). In practice most available MDDUI processes offer entry-points but implement forward engineering only. Thus, for instance in the SEESCOA approach [41] the abstract user interface is the entry point to a forward-engineering process, whereas in TERESA [49] two different entry points are available: The first one at the concepts-and task model and the second one at the abstract model level.

Development by Examples

Another approach two reduce the design complexity is to design model abstractions based on concrete instances of the system by following a test-driven development approach. In such an approach the designer creates examples after every transformation to test if the actual design model is able to capture the original requirements. Since by following a transformal development approach more concrete models are based on a transformation of abstract ones, examples that have been created for a more abstract model can be reused for testing subsequent models as well [28].

After I have introduced the basic models and processes lets take a look at the second general effort in MDDUI: the focus on tool-driven design of models.

7.2.4 Tool-driven model design

Since the development of tools for model-based user interface engineering has been done since the beginnings of MDDUI, there are two much tools available to discuss them all. Tool-development has been done in two different types of categories: First comprehensive development environments have been realized. These environments tightly integrate several tools with the advantage of offering a constant mode of operation that guides a developer through all models and transformations up to the final executable user interface. Second, tool-chains have been introduced that are based on standardized exchange formats that are defined for all ontological and archetypal models. Hence, various tools, each offering the design of an archetypal model or a transformation can be chained and selected based on the requirements of the project.

For the sake of brevity I focus on giving examples for both approaches in order to discuss the pros and cons of both approaches.

Integrated Development Environments

The Model-Based Interface Designer (MOBI-D) [56, 58] was one of the first proposals for an integrated model-based interface development environment that embraces the idea of designing and developing user interfaces by creating interface models. MOBI-D included model-editing tools, user-task elicitation tools, and an interface building tool called Model-Based Interface Layout Editor (MOBILE) [57]. MOBILE includes a knowledge-based decision support system to assist the designer. By a knowledge base of interface guidelines an inference mechanism traverses the attributes and objects of the domain model to propose suitable widgets to the designer.

The Transformation-based Environment for Designing and Developing Multi-Device Interfaces (TERESA) [6, 47, 48] is composed of a method consisting of four steps that are supported by one tool. First, by high-level task modeling of a multi-context application, a single task model is designed that addresses all possible contexts of use, the involved roles, as well as identifies all objects of the domain relevant for performing the tasks. Second, a system task model is developed for all the different platforms that should be supported. The system task model for a certain platform is allowed to refine the task model of the first step. Third, from the system task model an abstract user interface is derived that considers the temporal relationships and composes a set of presentations. Each presentation is structured by the means of interactors. Finally, by a user interface generator the final user interface presentation is generated for all platforms. The TERESA tool supports all these steps and interactively guides the

developer through this top-down approach and offers a set of transformations to generate voice and web-based applications as well.

The Dygimes system [39, 22] (Dynamically Generating Interfaces for Mobile and Embedded Systems) follows the same approach as TERESA and uses a task tree as the initial model to calculate the sets of enabled tasks (ETS). Based on the ETS a dialogue model is derived that forms a state transition network [40]. In Dygimes a tool supports the designer to attach XML-based user interface descriptions to all atomic tasks. After the mappings between the task tree and the user interface fragments have been specified, Dygimes offers a tool to describe spatial layout constraints helping the designer to ensure that the interface is rendered in a visually consistent manner. Finally, a run-time library can read the constructed UI specification, which includes the task specification and adapts both to the target platform and renders the user interface based on the calculated ETS.

The Dynamic Model-based User Interface Development (DynaMo-AID) project [17] is based on an extended version of CTT task models and supports dynamic context changes. They propose decision nodes and collect distinct sub-trees from which one will be selected at run-time. Further on, DynaMo-AID includes a concrete presentation model as well as a description of domain objects. Up to now it supports to generate an application based on UIML [1] and SEESCOA XML [4]. DynaMo-AID follows a prototype-driven methodology [20] that is inspired by the spiral model. It starts from a task model, followed by the generation of a dialogue model. Combined with the concrete presentation model a prototype can be generated, evaluated and refined. [3] criticizes that no details are available on the exact notation used by the DynaMo-AID tool for the textual description of the model.

The user interface generation process in DiaTask [59] works similar to DynaMo-AID. It starts with the specification of a task model, which is transformed to a dialogue graph. It's up to the designer to relate tasks with user interface elements that are represented using XUL as the concrete user interface language.

Tool-chaining

Vanderdonkt et al. have developed a whole bunch of tools [44, 21, 24, 46] that are based on a language (The USerInterface eXtensible Markup Language - UsiXML) making the models and their associated transformations explicit.

UsiXML is aimed at offering a comprehensive approach for describing the various levels of details and abstractions of a user interface depending on the context of use. Therefore it is structured following the basic abstraction levels offered by the Cameleon Reference Framework.

At the task- and concepts-level UsiXML [36, 37, 35] offers a task and a domain model that describes the objects and classes that are presented and manipulated by the user. At the abstract level, UsiXML's abstract user interface model specifies the groups of tasks and domain concepts that should be presented together at the same time to the user. Finally, the concrete user interface model is used to give a detailed specification of the user interface appearence, that is dependent on a certain modality (for instance to generate a graphical or a voice-based user interface).

Different to earlier approaches, that mainly focus on declarative descriptions of models on certain abstraction levels (like UIML[1] or XUL[2]), UsiXML explicitly considers two additional aspects: the context of use and support for a transformational development approach.

The context of use is considered by specifying three additional models: the user model that decomposes the users of the interactive system into stereotypes and describes them by attributes for instance to express their experience with the system or a specific task or their motivation. The environment model describes the global environment where the interaction between the user and the system takes place. Properties of the environment can be physically to describe the level of noise or the lightning, or psychologically to express the level of stress of the user. Finally the platform model is used to specify the relevant attributes of the hardware and software of the device that is used to present the user interface.

The transformal development support of UsiXML is based on graph transformations and is organized by graph rewriting rules equipped with negative application conditions and attribute conditions. This transformation can be interactively constructed or manipulated using an Editor [65] or can be processed automatically. Up to now, the transformations are used between the task and concepts level and the abstract user interface level, and between the abstract and the concrete user interface level.

7.3. The pros and cons of Multi-Targeting

Several pros and cons for following a model-based development approach in user interface design have been identified and intensively discussed in the recent years. The following section introduces the major advantages and actual shortcomings.

7.3.1 Advantages of transformational MDDUI

Compared to “classical” interface development performed with interfaces builders, the transformational development has advantages in methodology, re-usability, and consistency.

Methodology

Model-based approaches are driven by specifications that are subsequently derived by a predefined process. Starting the development cycle with a specification is a widely accepted software engineering principle as [32] notes. User-centered and user interface-centered development life cycles are supported. They let designers work with tasks, users, and domain concepts instead of thinking in engineering terms [68]. These models encourage to think more about artifacts that should be realized and force the designers to explicitly represent the rationale of design decisions [68].

Relying on declarative models is a common representation that design tools can reason about to criticize designs and to detect questionable features [11, 12]. Declarative models enable realizing automated advisers that can support the designer to refine the designs. Further on user interface construction tools can be based on declarative models

[1] http://www.UIML.org, last checked 8/8/2010

[2] https://developer.mozilla.org/en/xul, last checked 8/8/2010

that enable automated creation of portions of the user interface. During run-time a declarative representation by models can be used to generate context-sensitive help to assist the user like proposed in [66].

An interactive system specification by using models enable executing the system before all details of the user interface have been designed to enable early experiments with designs by an iterative development process before considerable coding effort has been spent [68].

Re-usability

For multi-platform development of user interfaces and user interface support for context dependent adaptations model-based tools can provide the fundament for an efficient development life cycle that offers an automatic portability across different devices. Furthermore the complete description of the whole interface in a declarative form allows reusing the most interesting components.

Consistency

Consistency is the big issue that needs to be guaranteed between the user interfaces that are generated for different target platforms. Model-based approaches foster consistency since user interfaces are systematically derived by a well structured transformal process. Since the final user interfaces share abstractions – at least at the initial design model (like the task- or the abstract user interface model) consistency between different FUIs is improved.

7.3.2. Disadvantages of transformational Development

Several shortcomings have been identified so far that need to be tackled down. Commonly cited are [55, 67, 50]: The high threshold, Unpredictability, the problem of propagating modifications, their proprietary models, the efficiency and performance. Further on transformational development has not targeted real multimodal systems and only support pre-defined context of use adaptations. In the following paragraphs I present the main downsides of multi-targeting to motivate the next chapter the presents model-driven run time environments to tackle some of these disadvantages.

High threshold

The high threshold of model-based approaches is one of the big issues that need to be solved to get a broader acceptance. Up to now, developers need to learn a new language in order to express the user interface specification. Thus, model-based approaches require models to be specified in special modelling languages and therefore require a form of programming that is not suitable for many interface developers or designers.

Design tools that enable visual programming by abstracting from a certain user interface language and are integrated in widely deployed development environments are a solution to lower the threshold for model-based approaches.

Unpredictability

Each abstraction by a certain model requires the designer to understand and think in the same abstractions of the model that is utilized. The higher the abstraction that the model offers compared to the final user interface, the harder it gets for the designer to understand how the model specifications are connected to the final user interface.

[30] proposes to rely on explicit transformation rules with a tool-based preview to reduce the unpredictability of model-based approaches. [35] applied such a graph-based transformational approach, but the pure amount of transformations to map between the various models of abstraction figured out hard to overview and maintain. Additionally, the wide range of user interface element concepts of the various platforms that need to be covered by the model-transformations is complex to handle and selecting the appropriate transformation between several alternatives (often there are several ways to realize a user interface) emphasizes as a difficult and challenging task.

Propagation of modifications

Supporting several models of abstraction for the design and specification of an interactive system includes allowing changes to each of these models later on. These changes need to be propagated to the other models to maintain consistency between all models. Whereas it is consent that support for abstract-to-concrete and concrete-to-abstract (reverse engineering) as well as various entry points should be supported by model-based approaches, tools and approaches that realize these options are still missing. [30] describes the propagation of modifications as tricky, but proposes to determine the side effects on the other models entailed by the application of rules.

Proprietary models

Most model based approaches have been strongly tied to their associated model-based system and cannot be exported or are not publicly available. UsiXML [36] was the first completely available model format description. Whereas it is currently target to standardization efforts, there is still a generally accepted set of model abstractions missing. Further on, model syntax has been explicitly specified in the UsiXML specification documents, but clear model semantics have not been sufficiently specified so far.

Efficiency and Performance

Efficiency and performance of model-based systems are rarely considered. Efficiency needs to be measured for the development cycle implementing multi-platform user interfaces. Performance has to be evaluated at run-time to test if the various supported adaptations do not restrict the user's performance.

Missing support for Multimodality

Multi-targeting is focused on generating isolated user interfaces by a step-by-step for different platforms or modalities (in practice for a specific markup or programming language), which is different to the generation of multimodal user interfaces that require a strong connection between the different interfaces for the connected modalities. The Cameleon Framework supports translations for user interfaces to adapt to new contexts at run-time but still misses the possibility to interconnect different modalities for multimodal fusion. [64] solves this by adding mappings for supporting synchronization between Voice CUI and HTML CUI, which has the disadvantage that because of the late introduction of the mappings for intermodal synchronization this could and up in a cumbersome task for huge applications, because each single element of the CUI has to be addressed manually by the developer to support synchronization.

Predefined Context of use

Whereas model-based approaches introduce various abstractions to design interactive systems, each enabling a comprehensive view of the whole application, they are currently focused on analysis- and design support. Thus, these approaches require the developer to consider all combinations of devices, which the interactive system should be able to adapt to during the development process. Each new mix of devices and modalities requires going through most of these design models again to compile a new user interface.

7.4. Beyond Multi-Targeting – Seamless Interaction in Smart Environment

In contrast to current PC-based systems, user interfaces for smart environments have a stronger need for integration and the adaptation to the usage situation, comprising the available devices, the situation of the user and other context information like e.g. light and noise level of the surrounding environment.

The tight integration of user interfaces in the environment enables seamless services that are ubiquitous available and are able to accompany the user through his daily life in a smart environment. Therefore these services are able to adapt their behavior and presentation continuously to maintain there usability by switching and combining devices and interaction modes as well as transforming their presentation to reflect context of use changes.

So far several characteristics of seamless services in smart environments have been identified [7] that are introduced in the following section.

7.4.1. Characteristics of Seamless Services for Smart Environments

Plasticity

The term Plasticity is inspired by the property of materials that are able to expand and contract under natural constraints without breaking and provide continuous usage. The term has been first introduced and applied to HCI by Calvary [15, 14]. In HCI it describes the capacity of an interactive application to withstand variations of context of use while preserving usability. This also covers the contraction and expansion of the set of tasks in order to preserve usability. Plasticity is similar to multi-targeting since both terms address the diversity of context of use adaptations, but additionally express requirements in terms of usability.

User interfaces for smart environments much more rely on the possibility to adapt to the context of use. This on the one hand is due to the fact that interaction happens in various situations and under different circumstances, as well as due to the fact that multiple different devices might be used for the interaction.

Multimodality

A multimodal interface is able to “process two or more combined user input modes – such as speech, pen, touch, manual gestures, gaze, and head and body movements – in a coordinated manner with multimedia system output.” [52] The multimodal form of communication with a computer allows recognizing naturally forms of human language and behavior and is driven by the idea to support more transparent, flexible, efficient, and powerfully expressive means of human-computer-interaction [52].

There are four different relations that describe how modes can be combined based on the interaction technique that they offer: Equivalence, Assignment, Redundancy, and Complementary (the CARE properties) that have been defined in TYCOON [42].

Equivalence describes a combination of modalities in that all can be used to lead to the same desired meaning, but only one modality can be used at a time. Thus, a text input prompt can be for instance either handled by a spoken response or by using a written text typed by a keyboard.

By *assignment* a particular modality is defined to be the only one that can be used to lead to the desired meaning. (e.g. a car can only be controlled by a steering wheel).

Modalities can be used *redundant* if they can be used individual and simultaneously to express the desired meaning. Hence, for instance speech and direct manipulation are used redundantly if the user articulates “show all flights to São Paulo” while pressing on the Button “São Paulo” with the mouse.

In a *complementary* usage of modalities, multiple complementary modalities are required to capture the desired meaning. For instance a “put that there” command requires both speech and pointing gestures in order to grasp the desired meaning.

Nigay et. all [51] have detailed these relationships between modalities and applied to tasks, interaction languages and devices. In their understanding a physical device can be used to issue or present physical actions to the user (e.g. pushing a mouse button or uttering a sentence). These actions a referred to from symbols and expressions of an interaction language. An interaction language “is a language used by the user or the system to exchange information” and “defines the set of possible well-formed expressions […] that convey meaning”.

This allows a more formal definition of these relations and defining the *equivalent* and *assignment* relations as *permanent* if they hold over any state to reach a goal and as *total* of they hold for all tasks that a user can perform with a system.

The realization of redundant and complementary modality compositions is complex, since they require fusion mechanisms to process the different input received from the complementary modalities to grasp the desired meaning. There are two main types of fusion: First, by *early fusion* (or micro-fusion) signals can be integrated at a feature level and second information can be merged at a semantic level, which is referred to as *late fusion* [52]. Early fusion is more appropriate for closely temporally synchronized input modes, where one mode influences the recognition in the others. An example for early fusion for redundant modes is combining lip movements with speech recognition. Late semantic fusion (or macro-fusion) is often applied for processing less coupled temporary information, such as speech and pen input.

There are two additional challenges in multimodal fusion, the temporal distance of incoming data and the difference in input data structure if different types of modalities are used and are discussed for instance in [27].

To prevent misunderstandings and interpretation often a high degree of redundancy for multimodal interaction is chosen. But Oviatt pointed out in [52] that the dominant theme in users’ natural organization of multimodal input actually is complementary of input. Even during multimodal correction of system errors, where users are highly

motivated to clarify and reinforce their information delivery they express redundant information less than 1% of the time (tested for speech and pen input [52]).

Session Persistence

Users tend to follow various tasks in parallel. Whereas some tasks can be accomplished in short term, long term tasks might be interrupted by more important ones and continued later on. This requires the handling of interruptions and the possibility to continue a service later on. Session Persistence has been used for instance for handling database connections (and for data persistence) and for managing communication in stateless protocols like HTTP. Different to these technical processes of maintaining a network connection or an application state, user interface session persistence includes identifying a user and enabling him to stop an interaction anytime without loosing the application as well as the interaction state. Stopped sessions can then be continued on any devices and with any mode later on.

Migration

Different to a stationary PC setup, in a smart environment the user is able to move around his environment while interacting with an application. Beneath mobile devices that are continuously available to the user, other (stationary) devices that are part of the smart environment network like for instance televisions, stereos or cameras can be seamlessly integrated while the interaction takes place. Therefore a user interface requires supporting migration. [2] distinguish between *total* and *partial migration*. *Total migration* describes the capability of interfaces to move to another platform while preserving their actual state. Thus, the user can comfortably continue the interaction exactly at the point where it has been stopped on the previous platform. Migration needs to consider a switch in modality as well. If the target platform offers for instance voice recognition instead of a keyboard to enter text, the text input from the user can be spoken where it has been stopped after the user has left the PC.

Partial migration describes the distribution of parts of the interface to several platforms and modalities and is more complex to handle from both, the user's as well as the developer's perspective. For the user the interface has to include tools to select which parts of it should be distributed and offer help to indentify suitable target platforms and modalities [62]. From the developer's perspective the complexity is about to handle the composing of partial distributed interfaces to a new one. Composing is required if at least two parts of two different interfaces are targeted to run on the same platform/modality combination and can includes merging of elements to eliminate duplicates. Control interfaces are a typical example for partial user interface distribution where merging can be performed. In the iCrafter service framework [54] for instance interfaces control appliances of a presentation room can be aggregated by merging elements. There, in a presentation scenario that includes turning off several lights, all individual light user interface controls can be merged based on service interfaces that implement "PowerSwitchInterface" profiles.

Similar to classical window managers or operating system frontends that offer a set of basic user operations and forms of organization driven by the WIMP (Windows, Icons, Menus and Pointer) interaction style and direct manipulation, a frontend to the user for smart environments relies on these characteristics that I have presented in this section.

These characteristics have been utilized to realize run-time environments, which allow the creation of user interfaces beyond multi-targeting and enable their incorporation in different applications. In the following section I will present the actual state of run-time-systems supporting embedding interfaces in the environment.

7.4.2. Model-driven Run-time-Environments (MRE)

In smart environments the basic challenge for an interface is to withstand the continuously happening changes of the context of use: Users are no longer sitting in front of a stationary PC but are moving around the environment and switching between devices and modalities while interacting. This initiates a new level of comfort in human-computer-interaction – the user can now focus more on her tasks since restrictions like the limited interaction capabilities of a personal mobile device or by the fixed position of a desktop PC do no longer exist in such environments. Hence, the technical challenge is no longer about multi-targeting which can address the *predictive contexts of use* (that can be foreseen at design-time) but on interfaces that are prepared to consider the unpredictable context, which is called the *effective contexts of use* that really occur at run-time.

Current run-time systems tackle this challenge by not only describing the static structure and the behavior of the interface, but although include a description of the evolution capabilities in the models. As an example I present two different approaches, an *interactor-centric* and a *model-centric approach* for realizing such a run-time environment. The former one puts a strong focus on a modularized way of constructing interfaces through widgets and export the evolutionary part of the models into a semantic network. The latter one maintains the principle design models of multi-targeting but embeds the behavior as well as a description of the evolution capabilities into the (design) models. Both approaches add an extension mechanism that enables to introduce new adaptations and interaction capabilities to the models or the semantic network respectively.

Additionally the user in a smart environment gets much more in focus. Whereas for multi-targeting, the user was mainly the target of the design (for instance by a user-centered design process), in a smart environment the user can manipulate or change the system’s interaction behavior based on his preferences. Therefore run-time systems include a meta user interface enabling the user or a designer to inspect and manipulate the run-time state of the system.

The **COMETs approach**

The Context Mobile Widget (COMET) architecture style [26] offers self descriptive interactors that support plasticity. Thus, COMETs are able to publish their quality in use that they can guarantee for a set of contexts of use.

Following the COMETs architectural style, the user tasks of a task decomposition are reflected by individual COMETS that can be grouped to build a presentation for a task by recursively composing them of other COMETs. In such a graph every child COMET expresses itself with regard to its parent. Additionally COMETs can be defined based on task operators such as for instance the interleaving operator to render COMETs e.g. side-by-side.

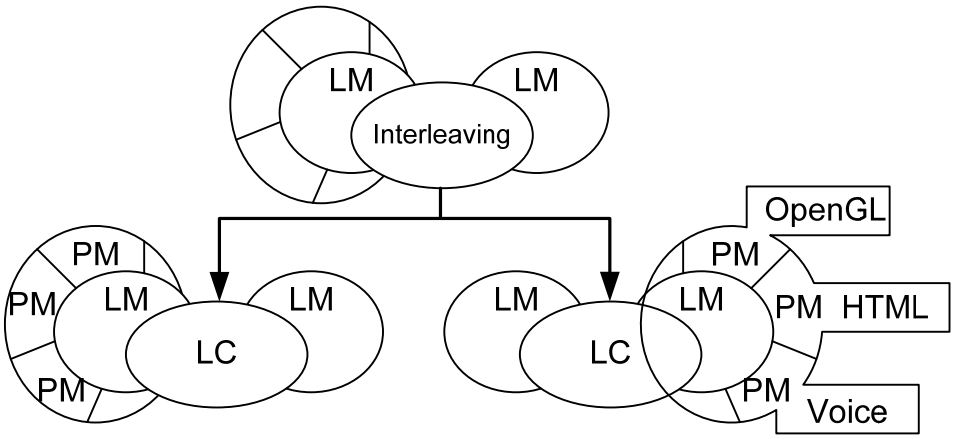


Figure 2: Graph of COMETs

Like illustrated by figure 2 each COMET is composed of three facets: 1) a logical consistency (LC), 2) a set of one or more logical models (LM) that the LC is associated to, and finally 3) a set of one or more physical models (PM) that are associated to a LM. Even if there is no fixed rule how to use these facets, the tasks, AUI, CUI, and FUI can be embodied in LCs, presentation LMs, PMs, and in technological primitives that target different languages and toolkits like for instance HTML or OpenGL respectively.

So far COMETs support the redundant and equivalent relations of the CARE properties by event propagation inside a COMET. By a domain specific language (COMET/RE) incoming events from one PM (e.g. graphics) can be specified to only get propagated by the LC to the other PMs if there is a redundant event incoming from another PM (e.g. voice).

At system run-time COMETs instantiate entities from a semantic network that describes the concepts of the different models of the CAMELEON framework like tasks, abstract containers, or concrete list boxes. The concepts of the network are structured with multiple types of relationships like for instance inheritance, abstraction, or composition.

If a graph of COMETs at run-time requires adapting to a new context of use an adaptation goal is specified and the semantic network can be queried to retrieve potential actions to transform the presentation.

The general advantage of the COMET approach lies in the fact that a COMET application can be easily extended to support new context of use adaptations by enhancing the semantic network without modifying the rest of a COMETs-based application. From a user's perspective the interactor-driven architecture has the advantage that every COMET directly expresses elements of the interface that can be directly manipulated by him (by specifying adaptation goals).

The Multi-Access Service Platform (MASP): Dynamic Executable Models

In contrast to the static design models of a multi-targeting approach, providing (only) a snapshot of the system under study at a given point in time, executable models provide the logic that defines the dynamic behavior as part of the model. The general idea is to equip executable models with "everything required to produce a desired functionality of a single Executable Models for Human-Computer Interaction problem domain" [43]. Hence, to construct an executable model, three capabilities have to be included: Beneath static element structures, the behavior, and the evolution of the system.

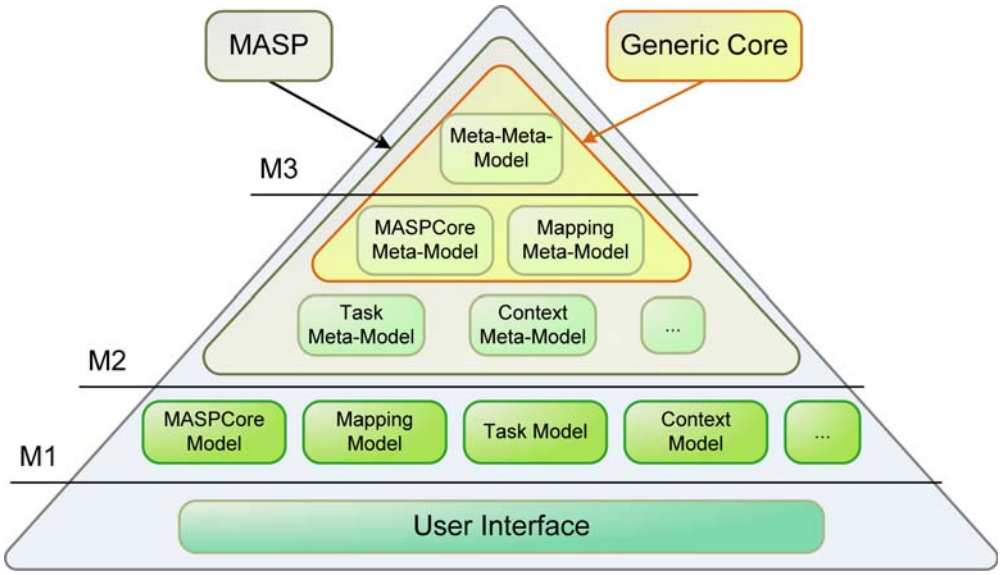


Figure 3: Executeable Models of the MASP in relation to the MOF Meta-Pyramid. Taken from [8]

Figure 3 illustrates the MOF Meta Pyramid in relation to the MASP approach that implements executable models. The core of the MASP offers an extension mechanism by a meta-meta model that distinguishes between three basic elements: definition-, situation- and execution elements that generalize all executable models.

The static structure of a design model is defined by definition elements stating all constants of a system that are not changing over time. Situation Elements define the current state of a model and capture all information that can change over time. Situation elements can trigger execution elements that are able to process and update the data of other situation elements.

The separation by the MOF model offers clear boundaries in the MASP during design-time: An application designer works with definition elements only, whereas a system architect modifies or enhances the run-time system by manipulating the meta-models of the M2 level (figure 3) to introduce for instance a new kind of task modeling or a next context description.

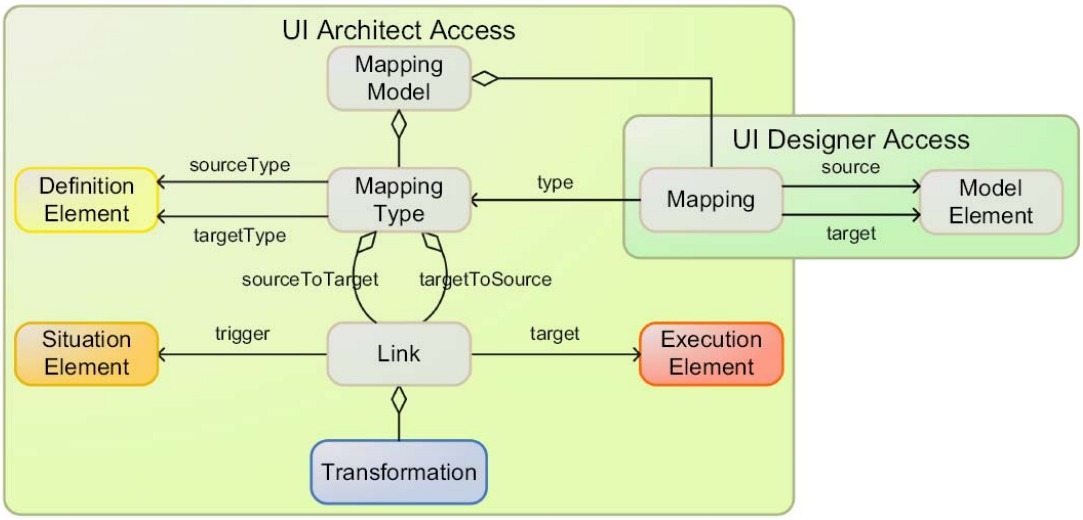


Figure 4: The Mapping Meta Model. Taken from [8].

Figure 3 depicts the mapping meta-model that connects multiple executable models in the MASP. It allows defining relations between their elements based on the structures given by the meta-meta-model and is therefore located at M2 layer of the MOF architecture. Defining the mappings in a separate model removes the problem of hard-coding them into the other models

The meta-mapping model specifies both, *mapping types* and the *mappings* that are used to synchronize elements of different models of the M1 layer, which define a particular application.

Like illustrated in figure 4, a *mapping type* refers to two definition elements that should be kept in sync. Therefore it contains *links* that get triggered by situation elements and call execution elements that process the data of the situation element. For transforming the situation element's data in an appropriate format to be processed by the execution element, a transformation can be specified. An example for such a mapping type definition is for instance the information gained from a task model, that an interaction task has been performed successfully by the user and that the interface presentation should be updated. Different to the mapping type, a concrete mapping reference a mapping type and relates it to actual application. Hence, an application developer can browse through the set of available mapping types and can create new mappings that reference the available mapping types. To implement such a mapping, the developer therefore has to set the specific source and target model elements to the mapping.

Discussion

The basic advantage of an interactor-centric approach is that interfaces can be composed by the same metaphors that are well known through interface builders: by drag-and dropping widgets on to the interfaces as well as positioning them based on the individual preferences of the user. For the user these widgets are seen as a black-box supporting the plasticity mechanisms internally. The model-centric architecture style abandons the functional grouping into separate interactors and directly executes the models of an MDDUI approach. A model-centric architecture can be directly deployed with the design models that have been developed following MDDUI as models are the

executing units instead of interactors. Since executable models just add further information like describing its behavior, already existing design models could be enhanced and already existing design tools for multi-targeting can still be used.

Meta user interfaces

Controlling services and their interfaces in a smart environment is a critical aspect in smart environments. They provide access to various services from numerous networked interaction devices for multiple users that accessing services in many different situations and via diverse combinations of devices and modalities.

Meta-user interfaces (meta-UI) are a special kind of end-user development environment that offer a set of function to control and evaluate the state of an interactive ambient space. This set is meta- because it covers all domain-dependent services that support human activities and is UI oriented because of its role enabling the user to control and to evaluate the ambient space [23].

Figure 5: Meta-UI Implementation of the MASP taken from [61].

Figure 5 depicts the meta-UI of the MASP with a running service interface. So far, the meta-UI offers four generic services that are implemented by the button bar at the bottom of the meta-UI: modality-, migration-, adaptation-, and distribution-configuration. By the modality configuration a user can set the utilized modalities. This is currently limited to the definition of equivalent related modalities, but helps the user to prevent unwanted side effects, like disabling a speech-recognition in noisy

environments or during a phone call for instance. The actual activated modalities are signaled by the status bar at the top of the meta-UI.

The migration and distribution configurations enable total respectively partial migration of interfaces. The migration is not limited to intra-modal remolding but supports a free selection of equivalent modalities for the target platform(s) of the migration. Finally with the adaptation functionality mixed-by-design objects [23] can be specified. Mixed-by-design objects couple physical entities with digital services that have been assembled during design-time. One such mixed-by-design object that is used in the MASP meta-UI is an active RFID tag that is coupled to a follow-me digital service. Activating this service results in a user-interface that continuously remolds to follows the user through the environment.

The COMETs Inspector implements such a meta-UI as well. Different to the MASP meta-UI that is so far limited by the mapping types that have been defined at system design-time, the COMETs inspector can be used to query the semantic network that stores and relates all instantiated interactors on all levels of the CAMELEON framework. Thus, it supports inspecting the current state of the system as well as issuing queries to the semantic network to retrieve potential transformation options. As stated in [25] the inspector is actual limited to basic operations like adding, removing or substituting elements, but coupled with the querying option to the semantic networks a first step to end-user development has been done since in principle transformation recipes and alternatives can be queried and applied by the user by this basic operations.

For a detailed discussion and characterization of meta-UIs based on a taxonomy I recommend [23].

7.5. Challenges for realizing Interfaces for Smart Environments

7.5.1 Modeling Real and Flexible Multimodal Interaction

Nowadays multimodal systems that support the user by a combination of speech, gesture and graphical-driven interaction are already part of our everyday life. Examples are combinations of speech-driven and graphical interfaces like in-car assistance systems, language-learning software, or tourist information systems. Recent video-game consoles just introduced gesture-based interactions like for instance the Nintendo Wii game console. Games can be controlled in a more natural and intuitive way by using hand gestures, balancing and moving the body. The market success of this console demonstrates that even new audiences can be addressed by enabling multimodal interaction to ease the usage of interactive systems.

In the recent years of research on multimodal systems there has been great success in demonstrating that one big research issue, the “multimodal fusion”, that is often motivated with the “put this there” example [9], can be handled. This example describes the problem of interpreting a user interaction spanning over several modalities (which requires the machine to combine the gesture, voice and graphical inputs of the user with his actual context to catch the user’s goals). In Germany for example, the SmartCom project [33, 73] realized such a system by using the blackboard metaphor to implement an interactive avatar that can interpret and react to multimodal user interaction. The multimodal fusion of speech, gestures, graphical inputs, and users’ emotions is driven by a standardized information exchange language (M3L) to enable the fusion of (uncertain) information of the different modalities that are connected to a multimodal setup. Beneath this large scale system, recent research combines component repositories containing re-usable device drivers and fusion mechanisms with interactive design tools to create and prototype various multimodal setups. One such example that receives a lot of attention recently is the Open Interface Platform [34] another popular one is iStuff [60].

This recent research results allow to rapidly creating sets of very different and even very specialized multimodal setups realizing control interfaces with the help of interactive editors. But so far they lack of efficient software engineering methods and notations to support the application design of multimodal interaction for such heterogeneous types of interaction devices and modalities.

Modeling multimodal systems that support various multimodal setups is an open research issue. A recent promising work by [64] that implemented a model-based development process to generate multimodal web interfaces stated the importance of considering the CARE-properties, but neglected the support of modeling complementary or redundant multimodal interaction to support multimodal fusion. UsiXML [37] and TERESA [5] do not offer a dialogue model but distribute it to several models (mainly the task model that specifies the interaction sequence and the abstract user interface model containing the navigation). Since these models are considered modality independent, supporting different navigations based on the used combination of modalities is difficult to implement. Based on the modality (e.g. tactile, speech or graphics (large screen vs. small screen with pagination) or the composition of modalities the navigation needs to be realized completely different.

7.5.2. The Mapping Explosion and Modeling Tool Problem

The mapping problem can be seen as a direct consequence following a model-based user interface development approach. The mapping problem has been firstly introduced by Angel Puerta and Jakob Eisenstein [58] who state: ”if for a given user interface design it is potentially meaningful to map any abstract model element to any concrete one, then we would probably be facing a nearly insurmountable computation problem”. Figure 3 illustrates the mapping problem between a set of models.

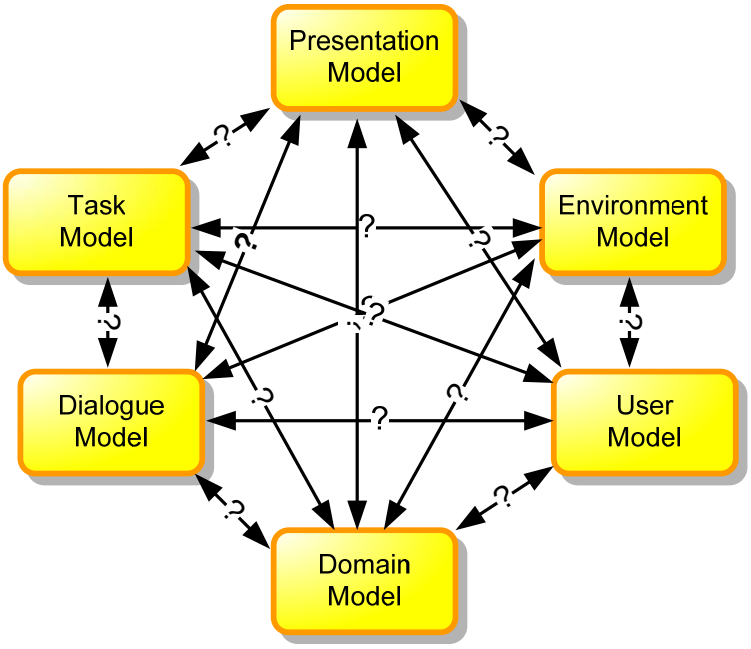


Figure 5: The mapping explosion problem

Solving the mapping problem comprises answering, which models should be combined, at which level of abstraction, and how the models are combined. With the introduction of run time systems that awake the original design models alive by making them executable, the original mapping problem explodes. Now there are two kinds of mappings that need to be specified: transformational mappings at design time and mappings at runtime that keep all running models in sync.

Like illustrated figure 3 the worst case would be to map every element of every model to every element of every other model. One could argue that by a transitional propagation the amount op mapping could be reduced. But there is an inherent performance requirement by the keep various modes and medias in sync at run time. Since each mapping typical requires transformational part that does the abstract-to-concrete (or vice versa) translation this can be challenging. To my knowledge so far performance evaluations of model-driven run time environments are missing.

There has been a lot of research about mapping mechanisms at design time that might be adapted to run time by an automated generation of runtime mappings based on the transformations that have been proposed at design time. For instance [71, 72] describe three mechanisms that can be used to establish mappings between the models: model derivation, model linking/binding, and model composition. [19] extended this

classification to five mechanisms: model derivation, partial model derivation, model linking, model modification, and model update.

Beneath the challenge to solve the mapping problem between the various models, so far the design of mapping is supported by tools only to a very limited extend - and so far only at design time. For instance [45] define a formal mapping between the domain, task and abstract user interface elements, which allows to establish a set of mappings either manually or automatically (in combination with the TransformiXML tool [65]). They offer eight different types of mapping relationships that can be defined.

Working with these transformational approaches based on mappings introduces two major problems. First, it is very formalized and programmatic approach that is hard to understand and to extend and second, supporting the flexible combining of modalities at run time for multimodal interaction would result in merging these mappings and requires them to interact which each other, which is considered even more complex to manage.

7.5.3 Participation of a Designer and End User Development

Important groups are not involved in current MDDUI design cycles: neither the end user as well as an interaction designer can participate actual MDDUI development processes. The end user suffers from complexity and the designer from gap in tools between those that support their workflow and those that are currently available in MDDUI.

Initial efforts have been done by realizing specialized tools for designers to include prototyping approaches on different levels of fidelity: For instance SketchiXML [24], a sketching editor is targeted to low-fidelity prototyping and replaces paper sketching of user interfaces, the VisiXML tool [70] supports vector drawing, whereas GrafiXML [36] is an advanced user interface editor supporting high-fidelity prototyping. One common aspect of all these prototyping tools is that they focus on the design of graphical user interfaces. Supporting interaction design for smart environments that includes considering context of use adaptation and flexible multimodal is so far not covered.

Further on, the fundamental challenge to develop environments that enable people without particular background in programming to realize their own interactive applications [53] still exists. Although first approaches based on the meta-UI paradigm have been proposed that I have presented earlier. These are targeted to enable end users to manipulate an interactive space and can be seen as an initial step to support end user development.

7.6. References

[1] Marc Abrams und Jim Helms: *User Interface Markup Language (UIML) Specification version 3.0*; Techn. Ber.; Harmonia Inc.; 2002.

[2] Renata Bandelloni und Fabio Paterno: *Flexible Interface Migration*; in *Proceedings of the 9th International Conference on Intelligent User Interfaces*; S. 148 – 155; Funchal, Madeira, Portugal; 2004; ACM Press New York, NY, USA.

[3] Jan Van den Bergh: *High-Level User Interface Models for Model-Driven Design of Context-Sensitive User Interfaces*; Dissertation; Hasselt University and transnational University of Limburg School of Information Technology; 2006.

[4] Jan Van den Bergh und Karin Coninx: *Model-based design of context-sensitive interactive applications: a discussion of notations*; in *TAMODIA '04: Proceedings of the 3rd annual conference on Task models and diagrams*; S. 43–50; New York, NY, USA; 2004; ACM Press.

[5] Silvia Berti, Francesco Correani et al.: *TERESA: A Transformation-based Environment for Designing and Developing Multi-Device Interfaces*; in *ACM CHI 2004, Extended Abstract*; Bd. II; S. 793–794; Vienna, Austria; April 2004; ACM Press.

[6] Silvia Berti und Fabio Paterno: *Migratory MultiModal Interfaces in MultiDevice Environments*; in *ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces*; S. 92–99; New York, NY, USA; 2005; ACM Press.

[7] Marco Blumendorf, Sebastian Feuerstack und Sahin Albayrak: *Multimodal Smart Home User Interfaces*; in *Intelligent User Interfaces for Ambient Assisted Living: Proceedings of the First International Workshop IUI4AAL 2008* (Hg. Kizito Mukasa, Andreas Holzinger und Arthur Karshmer). IRB Verlag; 2008.

[8] Marco Blumendorf, Grzegorz Lehmann et al.: *Executable Models for Human-Computer Interaction*; in *Interactive Systems. Design, Specification, and Verification: 15th International Workshop, DSV-IS 2008 Kingston, Canada, July 16-18, 2008 Revised Papers* (Hg. T. C. Nicholas Graham und Philippe Palanque); S. 238–251; Berlin, Heidelberg; 2008; Springer-Verlag.

[9] Richard A. Bolt: *"Put that there": Voice and Gesture at the Graphics Interface*; in *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '80)*; S. 262–270; New York, NY, USA; 1980; ACM Press.

[10] Laurent Bouillon, Jean Vanderdonckt und Kwok Chieu Chow: *Flexible re-engineering of web sites*; in *IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces*; S. 132–139; New York, NY, USA; 2004; ACM.

[11] Robert Eric Braudes: *A framework for conceptual consistency verification*; Dissertation; George Washington University; Washington, DC, USA; 1990.

[12] Michael D. Byrne, D. Wood et al.: *Automating interface evaluation*; in *CHI Conference Companion* (Hg. Catherine Plaisant); S. 216. ACM; 1994.

[13] Gaelle Calvary, Joelle Coutaz et al.: *A Unifying Reference Framework for Multi-Target User Interfaces*; *Interacting with Computers*; 15(3), S. 289–308; 2003.

[14] Gaelle Calvary, Joelle Coutaz et al.: *Towards a New Generation of Widgets for Supporting Software Plasticity: The "Comet"*; in *Proceeidings of Engineering Human Computer Interaction and Interactive Systems, Joint Working Conferences EHCI-DSVIS 2004. ISSN 0302-9743* (Hg. Remi Bastide, Philippe A.

Palanque und Joerg Roth); Bd. 3425 von *Lecture Notes in Computer Science*; S. 306–324; Berlin/Heidelberg; 2004; Springer.

[15] Gaelle Calvary, Joelle Coutaz und David Thevenin: *Supporting Context Changes for Plastic User Interfaces: A Process and a Mechanism*; in *Joint Proceedings of HCI'2001 and IHM'2001* (Hg. Jean Vanderdonckt A. Blandford und P. Gray); S. 349–363; Lille; September 2001; Springer-Verlag.

[16] Hao hua Chu, Henry Song et al.: *Roam, A seamless application framework*; *Journal of Systems and Software*; 69(3), S. 209–226; 2004.

[17] Tim Clerckx, Kris Luyten und Karin Coninx: *DynaMo-AID: A Design Process and a Runtime Architecture for Dynamic Model-Based User Interface Development*; in *The 9th IFIP Working Conference on Engineering for Human-Computer Interaction Jointly with The 11th International Workshop on Design, Specification and Verification of Interactive Systems (EHCI/DS-VIS), ISBN 3-540-26097-8*; Bd. 3425; S. 77–95; Springer, Berlin; 2004.

[18] Tim Clerckx, Kris Luyten und Karin Coninx: *Generating Context-Sensitive Multiple Device Interfaces from Design*; in *Pre-Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces CADUI 2004*; 2004.

[19] Tim Clerckx, Kris Luyten und Karin Coninx: *The Mapping Problem Back and Forth: Customizing Dynamic Models while Preserving Consistency*; in *TAMODIA* (Hg. Pavel Slavik und Philippe A. Palanque); S. 33–42; Prague, Czech Republic; 2004; ACM International Conference Proceeding Series.

[20] Tim Clerckx, Chris Vandervelpen et al.: *A Prototype-Driven Development Process for Context-Aware User Interfaces*; in *Proceedings of the 5th International Workshop,Task Models and Diagrams for Users Interface Design (TAMODIA 2006)*; Bd. 4385 von *Lecture Notes in Computer Science*; S. 339–354; Berlin / Heidelberg; August 2006.

[21] Benoit Collignon, Jean Vanderdonkt und Gaelle Calvary: *Model-Driven Engineering of Multi-Target Plastic User Interfaces*; in *The Fourth International Conference on Autonomic and Autonomous Systems (ICAS 2008)*; 2008; IEEE Computer Society Press, March 16-21, 2008 - Gosier, Guadeloupe.

[22] Karin Coninx, Kris Luyten et al.: *Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems*; in *Mobile HCI* (Hg. Luca Chittaro); Bd. 2795 von *Lecture Notes in Computer Science*; S. 256–270. Springer; 2003.

[23] Joelle Coutaz: *Meta-User Interfaces for Ambient Spaces*; in *TAMODIA* (Hg. Karin Coninx, Kris Luyten und Kevin A. Schneider); Bd. 4385 von *Lecture Notes in Computer Science*; S. 1–15. Springer; 2006.

[24] Vanderdonckt J. Coyette, A. und Q. Limbourg: *SketchiXML: An Informal Design Tool for User Interface Early Prototyping*; in *Proceedings of RISE 2006 Workshop on Rapid User Interface Prototyping Infrastructures Applied to Control Systems RUIPICAS 2006* (Hg. V. Amaral M. Risoldi); Geneva; September 2006 2006.

[25] Alexandre Demeure, Gaelle Calvary et al.: *The Comets Inspector: Towards Run Time Plasticity Control based on a Sematic Network*; in *TAMODIA '06: Proceedings of the 5th annual conference on Task Models and DIAgrams,ISBN978-3-540-70815-5*; Bd. 4385 von *Lecture Notes in Computer Science*; S. 324–338; Berlin / Heidelberg; 2006; Springer.

[26] Alexandre Demeure, Gaelle Calvary und Karin Coninx: *COMET(s), A Software Architecture Style and an Interactors Toolkit for Plastic User Interfaces*; S. 225–237; 2008; Design, Specification, and Verification, 15th International Workshop, DSV-IS 2008, T.C.N. Graham & P. Palanque (Eds), Lecture Notes in Computer Science 5136, Springer Berlin / Heidelberg, Kingston, Canada, July 16-18, 2008, pp 225-237.

[27] Bruno Dumas, Denis Lalanne et al.: *Strengths and weaknesses of software architectures for the rapid creation of tangible and multimodal interfaces*; in *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*; S. 47–54; New York, NY, USA; 2008; ACM.

[28] Sebastian Feuerstack: *A Method for the User-centered and Model-based Development of Interactive Applications*; Dissertation; Technische UniversitÃ¤t Berlin; 2008.

[29] Sebastian Feuerstack, Marco Blumendorf et al.: *Automated Usability Evaluation during Model-Based Interactive System Development*; in *HCSE-TAMODIA '08: Proceedings of the 2nd Conference on Human-Centered Software Engineering and 7th International Workshop on Task Models and Diagrams*; S. 134–141; Berlin, Heidelberg; 2008; Springer-Verlag.

[30] Murielle Florins: *Graceful Degradation: a Method for Designing Multiplatform Graphical User Interfaces*; Dissertation; Universite Catholique de Louvain; Louvain-la-Neuve, Belgium; July 11th 2006.

[31] Murielle Florins, Francisco Montero Simarro et al.: *Splitting Rules for Graceful Degradation of User Interfaces*; in *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*; S. 264–266; New York, NY, USA; 2006; ACM Press.

[32] Carlo Ghezzi, Mehdi Jazayeri und Dino Mandrioli: *Fundamentals of software engineering*; Prentice-Hall, Inc.; Upper Saddle River, NJ, USA; 1991.

[33] Gerd Herzog und Alassane Ndiaye: *Building Multimodal Dialogue Applications: System Integration in SmartKom*; in *SmartKom: Foundations of Multimodal Dialogue Systems* (Hg. Wolfgang Wahlster); S. 439–452; Springer; Berlin, Heidelberg; 2006.

[34] Jean-Yves Lionel Lawson, Ahmad-Amr Al-Akkad et al.: *An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components*; in *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*; S. 245–254; New York, NY, USA; 2009; ACM.

[35] Quentin Limbourg: *Multi-Path Development of User Interfaces*; Dissertation; Universite Catholique de Louvain, Institut d'Administration et de Gestion (IAG); Louvain-la-Neuve, Belgium; September 2004.

[36] Quentin Limbourg, Jean Vanderdonckt et al.: *USIXML: A Language Supporting Multi-path Development of User Interfaces*; in *EHCI/DS-VIS* (Hg. Remi Bastide, Philippe A. Palanque und Joerg Roth); Bd. 3425 von *Lecture Notes in Computer Science*; S. 200–220. Springer; 2004.

[37] Quentin Limbourg, Jean Vanderdonckt et al.: *USIXML: A User Interface Description Language for Context-Sensitive User Interfaces*; in *Proceedings of the ACM AVI'2004 Workshop" Developing User Interfaces with XML: Advances on User Interface Description Languages*; S. 55–62; 2004.

[38] L.Paganelli und F.Paterno: *A Tool for Creating Design Models from Web Site Code*; *International Journal of Software Engineering and Knowledge Engineering*; 13(2), S. pp. 169–189; 2003.

[39] Kris Luyten: *Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development*; Dissertation; Transnationale Universiteit Limburg, School voor Informatietechnologie; October 2004.

[40] Kris Luyten, Tim Clerckx et al.: *Derivation of a Dialog Model from a Task Model by Activity Chain Extraction.*; in *Interactive Systems: Design, Specification, and Verification, 10th International Workshop (DSV-IS)*; S. 203–217; Funchal, Madeira Island, Portugal; June 2003.

[41] Kris Luyten, Tom Van Laerhoven et al.: *Runtime transformations for modal independent user interface migration*; *Interacting with Computers*; 15(3), S. 329–347; 2003.

[42] Jean-Claude Martin: *TYCOON: Theoretical Framework and Software Tools for Multimodal Interfaces*; *Intelligence and Multimodality in Multimedia interfaces, AAAI Press*; 1998.

[43] Stephen J. Mellor: *Agile MDA*; Techn. Ber.; Project Technology, Inc.; June 2004.

[44] Vanderdonckt-J. Michotte, B.: *GrafiXML, A Multi-Target User Interface Builder based on UsiXML*; in *Proceedings of 4th International Conference on Autonomic and Autonomous Systems ICAS 2008*. IEEE Computer Society Press; 16-21 March 2008.

[45] Francisco Montero, Vctor López-Jaquero et al.: *Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML*; in *Proceedings of 12th Int. Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS'2005)*; S. 161–172; 2005.

[46] Francisco S. Montero und Victor Lopez-Jaquero: *Fast HI-FI prototyping by using IdealXML*; Techn. Ber.; Departamento de Sistemas Informaticos, Universidad de Castilla-La Mancha; March 2006.

[47] Giulio Mori, Fabio Paterno und Carmen Santoro: *CTTE: Support for Developing and Analyzing Task Models for Interactive System Design.*; *IEEE Trans. Software Eng.*; 28(8), S. 797–813; 2002.

[48] Giulio Mori, Fabio Paterno und Carmen Santoro: *Tool support for designing nomadic applications*; in *IUI '03: Proceedings of the 8th International Conference on Intelligent User Interfaces*; S. 141–148; New York, NY, USA; 2003; ACM Press.

[49] Giulio Mori, Fabio Paterno und Carmen Santoro: *Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions*; *IEEE Transactions on Software Engineering*; 30(8), S. 507–520; 2004.

[50] Brad Myers, Scott E. Hudson und Randy Pausch: *Past, present, and future of user interface software tools*; *ACM Transactions on Human-Computer Interaction*; 7(1), S. 3–28; 2000.

[51] Laurence Nigay und Joelle Coutaz: *Multifeature Systems: The CARE Properties and Their Impact on Software Design*; in *Intelligence and Multimodality in Multimedia Interfaces*; 1997.

[52] Sharon Oviatt: *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications, 2nd Edition*; Kap. Multimodal Interfaces, S. 413–432; Lawrence Erlbaum; 2. Aufl.; 2008.

[53] Fabio Paterno: *Model-based Tools for Pervasive Usability*; *Interacting with Computers*; 17(3), S. 291–315; 2005.
[54] Shankar Ponnekanti, Brian Lee et al.: *ICrafter: A Service Framework for Ubiquitous Computing Environments.*; in *Ubicomp*; S. 56–75; 2001.
[55] Angel R. Puerta: *The MECANO Project: Comprehensive and Integrated Support for Model-Based Interface Development.*; in *Computer-Aided Design of User Interfaces (CADUI'96)*; S. 19–36; 1996.
[56] Angel R. Puerta: *A Model-Based Interface Development Environment*; *IEEE Softw.*; 14(4), S. 40–47; 1997.
[57] Angel R. Puerta, Eric Cheng et al.: *MOBILE: User-centered interface building*; in *CHI '99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*; S. 426–433; New York, NY, USA; 1999; ACM Press.
[58] Angel R. Puerta und Jacob Eisenstein: *Towards a General Computational Framework for Model-Based Interface Development Systems*; in *IUI99: International Conference on Intelligent User Interfaces*; S. 171–178; New York, NY, USA; January 1999; ACM; ISBN:1-58113-098-8.
[59] Daniel Reichard, Peter Forbrig und Anke Dittmar: *Task Models as Basis for Requirements Engineering and Software Execution*; in *Proceedings of TAMODIA 2004*; S. 51 – 57; Prague, Czeck Republic; 2004; ACM Press.
[60] Meredith Ringel, Joshua Tyler et al.: *iStuff: A Scalable Architecture for Lightweight, Wireless Devices for Ubicomp User Interfaces*; Proceedings of UBICOMP 2002; 2002.
[61] Dirk Roscher, Marco Blumendorf und Sahin Albayrak: *Using Meta User Interfaces to Control Multimodal Interaction in Smart Environments*; in *Proceedings of the IUI'09 Workshop on Model Driven Development of Advanced User Interfaces*; 2009.
[62] Dirk Roscher, Marco Blumendorf und Sahin Albayrak: *A Multimodal User Interface Model For Runtime Distribution*; in *Proceedings of the CHI'10 Workshop on Model Driven Development of Advanced User Interfaces*; Atlanta, Georgia, USA.; 2010.
[63] N. Souchon, Q. Limbourg und Jean Vanderdonckt: *Task Modelling in Multiple contexts of Use*; in *Interactive Systems:Design, Specification, and Verification (DSV-IS) 2002*; Bd. 2545/2002; S. 59–73. Springer Berlin / Heidelberg; 2002.
[64] Adrian Stanciulescu: *A Methodology for Developing Multimodal User Interfaces of Information Systems*; Dissertation; Universite Catholique de Louvain; 2008.
[65] Adrian Stanciulescu, Quentin Limbourg et al.: *A transformational approach for multimodal web user interfaces based on UsiXML*; in *ICMI '05: Proceedings of the 7th International Conference on Multimodal Interfaces*; S. 259–266; New York, NY, USA; 2005; ACM Press.
[66] Piyawadee Noi Sukaviriya: *Dynamic Construction of Animated Help from Application Context*; in *ACM Symposium on User Interface Software and Technology*; S. 190–202; 1988.
[67] Pedro A. Szekely: *Retrospective and Challenges for Model-Based Interface Development*; in *DSV-IS 96: 3rd International Eurographics Workshop on Design, Specification, and Verification of Interactive Systems* (Hg. François Bodart und Jean Vanderdonckt); S. 1–27. Springer; 1996.
[68] Pedro A. Szekely, Piyawadee Noi Sukaviriya et al.: *Declarative interface models for user interface construction tools: The MASTERMIND approach*; in

Proceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction; S. 120 – 150; 1995; ISBN:0-412-72180-5.

[69] D. Thevenin: *Adaptation en Interaction Homme-Machine: Le cas de la Plasticite.*; Dissertation; Ph.D. thesis, Universite Joseph Fourier, Grenoble I; 2001.

[70] Jean Vanderdonckt: *A MDA-Compliant Environment for Developing User Interfaces of Information Systems*; in *CAiSE 2005: Advanced Information Systems Engineering.* (Hg. Oscar Pastor und Joao Falcao e Cunha); Bd. 3520 von *Lecture Notes in Computer Science*; S. 16–31. Springer; 2005.

[71] Jean Vanderdonckt, Quentin Limbourg et al.: *Using a Task Model to Derive Presentation and Dialog Structure*.

[72] Jean Vanderdonckt, Quentin Limbourg und Murielle Florins: *Deriving the Navigational Structure of a User Interface*; in *Proceedings of IFIP INTERACT'03: Human-Computer Interaction*; 2: HCI methods; S. 455; 2003.

[73] Wolfgang Wahlster: *SmartKom: Fusion and Fission of Speech, Gestures, and Facial Expressions*; in *Proc. of the 1st International Workshop on Man-Machine Symbiotic Systems*; 2002.

Capítulo

8

Garantia de Segurança em aplicações: De Requisitos a Padrões Seguros

Rodrigo Elia Assad, Felipe Silva Ferraz e Silvio Romeiro Lemos Meira

Abstract

It is historically known that extensive phases of rework affect directly both costs and quality of already developed applications. In regard to requirements, systems security figure among those which are left behind, generating undesired steps of reformulation. This article is a short presentation on design patterns and non functional security requirements; it relates both and shows that an initial use of certain patterns, in early development phases, might reduce the impacts of relegating the systems security requirement implementation to a second plan.

Resumo

Historicamente, sabe-se que extensas fases de manutenção ou alteração afetam diretamente os custos e a qualidade das aplicações desenvolvidas. Muitos destes problemas estão associados à negligência na elaboração dos requisitos, neste cenário Segurança dos Sistemas figura entre aqueles que mais são deixados em segundo plano, gerando indesejáveis etapas de reformulação. Por outro lado, quando se trata do envolvimento com Padrões de Projeto *esses aparecem como umas das principais respostas para a solução de problemas relacionados ao desenvolvimento. Diante disto, esse trabalho propõe uma relação entre Padrões de Projeto e Requisitos de Segurança de modo a, uma vez adotado nas etapas iniciais do desenvolvimento, diminuir os impactos relacionados à implementação de segurança de software a segundo plano.*

1.1. Segurança da Informação

O termo segurança é normalmente utilizado para definir técnicas utilizadas para minimizar as vulnerabilidades de bens e recursos, onde bens são qualquer coisa de

valor. Vulnerabilidade é qualquer fraqueza que possa ser explorada para se violar o sistema ou as informações que este contem ou os serviços que este provê[I. 7498-2, 1989]. Segurança depende mais do que apenas da integridade do software e dos mecanismos de proteção, ela também depende da própria configuração e uso do software[Zerkle, 1996] em si, segurança está também relacionada com a proteção aos acessos, manipulação intencional ou não de valores confidenciais armazenados no sistema bem como sua utilização não autorizada por terceiros.

Existem três conceitos básicos de relacionados a temas de Segurança de maneira geral, são eles: Confidencialidade, Integridade e Disponibilidade. Já os conceitos que estão relacionados diretamente com as pessoas que utilizam as informações ou serviços que os sistemas provêem são: Autenticação, Autorização e Não-Repudiação.

Quando uma informação é acessada ou copiada por alguém sem autorização para tal, o resultado é conhecido por perda de confidencialidade. Exemplos de bens que podem ser relacionados à confidencialidade podem incluir desde dados de pesquisa, como informações de usuários ou até mesmo valores bancários. Por outro lado, quando temos a alteração de uma informação disponível em uma rede de forma insegura ou em algum sistema inseguro, podemos ter que lidar com perdas de integridade que significa que modificações não autorizadas foram feitas na informação. Integridade é particularmente importante para setores como o financeiro, controle de trafego aéreo e controle financeiro. Por fim quando informações são apagadas ou se tornam inacessíveis, o resultado é a perda de disponibilidade, isso significa que pessoas autorizadas a ter acesso à determinada informação não conseguem acessá-la.

Diante desse cenário, tem sido firmada a existência de três etapas necessárias para se garantir a segurança de sistemas ou aplicações são elas: autenticação, autorização e auditoria[Samarati 2000].

Essencialmente essas etapas visão garantir que os princípios definidos anteriormente sejam respeitados e que dessa forma possa-se falar em segurança da informação. Cada uma dessas faz usos de uma, ou uma combinação, de métodos ou práticas para tentar garantir que o mal usou e ou acessos indevidos, não sejam executados. Entre eles podemos citar desde o estabelecimento de normas para senhas, como conceitos de criptografia moderna e até mesmo soluções de padrões de segurança.

1.1.1 Autenticação

Desde sua criação, e posterior popularização, a Internet sempre se mostrou com um mecanismo de anonimato, onde pessoas poderiam passar-se por outras ou até mesmo não passar por ninguém. Peter Steiner[Steiner 1993], Figura 1, satirizou essa situação com um cartoon que ficaria conhecido como uma das motivações para criação e definição de mecanismos onde esse tipo de situação pudesse ser combatido.

Figura 1: "Na Internet, ninguém sabe que você é um cachorro"[Steiner 1993].

Autenticação é o processo onde uma pessoa ou programa de computador prova a sua identidade com o intuito de obter alguma informação ou executar alguma ação[Kunyu 2009].

A identidade de uma pessoa é uma simples abstração, um identificador em uma aplicação específica, por ex. Provar, ou validar, consiste na parte mais importante desse conceito, é geralmente feita através de uma informação conhecida, como uma senha, ou palavra chave, ou um cartão com identificação visual, ou algo único com relação a sua aparência, impressões digitais, íris ou até mesmo amostra de DNA.

Um sistema de autenticação pode ser considerado forte, se fizer uso de pelo menos 2 dos 3 tipos de autenticações disponíveis.

A fraqueza desse tipo de sistema gira em torno da transmissão de tais informações através de canais, senhas podem ser roubadas, acidentalmente reveladas, ou até mesmo esquecidas.

1.1.2 Autorização

Autorização é o processo de dar permissão para fazer ou ter algo. Em Sistemas de múltiplos usuários um administrador de sistemas definirá quais usuários terão acesso e quais privilégios de execução esses terão[Schumacher 2006]. Considerando que alguém conseguiu se Autenticar em um sistema, o sistema pode necessitar identificar quais recursos a entidade poderá receber durante a sessão atual. O mecanismo de autorização é muita vezes visto como composto de dois momentos[Wang2009]:

- Ato de atribuir permissões ao usuário do sistema no momento de seu cadastro, ou posterior.
- Ato de checar as permissões que foram atribuídas ao usuário no momento de seu acesso.

Logicamente, o mecanismo de autorização vem posteriormente ao de Autenticação.

Assumindo agora que Autorização é no mínimo tão importante quanto Autenticação, faz-se necessário estender o escopo das soluções de segurança, que por ventura estejam relacionadas à Autenticação, para tentar solucionar também problemas de Autorização.

1.1.3 Auditoria

O processo de auditar um sistema é um dos mais importantes mecanismos de segurança que se pode fazer uso na hora de examinar, verificar e corrigir o funcionamento geral de um sistema. Conceitualmente o processo de auditar um sistema verifica se o mesmo está executando suas funções corretamente. Esse tipo de verificação é extremamente importante para processos de segurança, afinal não podemos considerar suficiente, apenas, implementar mecanismo de segurança em determinados ambientes, precisamos disponibilizar um mecanismo onde se possa verificar que os mecanismos realmente funcionam[Schumacher 2006][Dhillon 2008].

Existem duas etapas básicas de Auditoria, que na verdade, são apenas variações do mesmo conceito. A primeira forma trata do registro de mudanças no estado do sistema, ou seja, eventos e/ou mudanças de estados são registrados. A segunda forma seria um processo sistemático que examina e verifica o sistema, trata-se de uma consulta ao ambiente para avaliação do funcionamento do mesmo, ou seja, verifica-se se o sistema está funcionando de forma correta. Para um processo de auditoria ser eficiente, ambas as formas mencionadas precisam estar construídas e sintonizadas, é impossível executar uma avaliação sistemática de um sistema ou de um evento se esses não foram registrados, e mais, se os estados do sistema não forem gravados, não há a menor necessidade de se guardar, também, as mudanças no sistema, assim vamos considerar que para uma auditoria ser satisfatória esta deverá:

- Registrar os estados do sistema;
- Verificar, sistematicamente os estados armazenados[Dhillon 2008].

- Permitir que os registros sejam checados[Schumacher 2006].

Em linhas gerais, uma vez concluídas a etapa de Autenticação, onde se identifica o usuário e a etapa de Autorização, onde se atribuem às permissões do usuário, a etapa de Auditoria consiste em uma etapa mais longa, presente durante o ciclo de vida do sistema, ela vem para monitorar o uso do sistema, servindo, entre outras, como input para futuras ações, que podem incluir desde melhorias nas etapas anteriores até reestruturações do ambiente.

1.1.4 Não-Repudiação

Uma vez que temos um processo de Autenticação onde o usuário prova quem ele é, posteriormente temos o processo de Autorização onde este recebe suas permissões junto ao sistema, por fim temos os mecanismos de auditoria para monitorar o uso desse sistema registrando ações tomadas pelos usuários para posterior checagem. Sendo assim, uma vez validados o usuário e suas permissões e suas ações tendo sido registrada, não é factível que existem mecanismo que permitam a uma entidade autenticada negar suas ações. Nesse cenário temos o termo Não-Repudiação.

Não-Repudiação pode ser definido como um mecanismo ou serviço que provê provas da integridade e origem de um dado ambos de forma inegável e não criável, que pode ser verificado por uma terceira parte. Um serviço que pode atestar uma relação entre dado e autor do dado de forma genuína[Peris 2008][Adikari 2006]

1.2. Requisitos de Segurança e Padrões de Projeto

1.2.1.Padrões de Projeto e a Gangue dos Quatro

O entusiasmo, dos desenvolvedores de sistema, criado ao redor dos padrões é algo inquestionável desde a publicação do livro da conhecida, Gangue dos quatro ou Gang-of-Four. Desenvolvedores do mundo todo se uniram ao redor da idéia dos padrões na certeza de que esses permitiram uma maior clareza, versatilidade, direção e facilidade na escrita de códigos dessa forma, padrões conseguiram trilhar seu caminho para fazer parte de inúmeros projetos[Schumacher, 2006].

Para os autores do GoF[Gamma, 1994] padrões de projeto variam de acordo com a granularidade deles e nível de abstração, por conta dessa grande variação fez-se necessário organizá-los e categorizá-los. A classificação proposta auxilia no aprendizado do padrão e na forma de achar o padrão mais adequado para cada problema.

A classificação proposta levou em consideração dois critérios. O primeiro chamando propósito reflete o que o padrão faz. Podem ser de criação, estrutural ou comportamental. Padrões de criação tratam do processo de criação dos objetos. Padrões estruturais lidam na forma como as relações entre os objetos são criadas e por fim padrões de comportamento caracterizam a forma como objetos interagem e distribuem responsabilidades.

O segundo critério, chamado de Escopo, trata de onde o padrão é aplicado se a classes ou os objetos. Padrões de Classe lidam com relacionamentos entre as classes e suas subclasses. Padrões de Objetos lidam com relações que podem ser mudadas em tempo de execução e são mais dinâmicos.

Para os fins desse trabalho detalharemos mais os a classificação relacionada ao propósito.

1.2.1.1 Padrões de Criação.

Padrões de Projeto de criação são responsáveis por abstrair o processo de criação de determinados objetos. Auxiliam a deixar o sistema de forma independente de como os objetos serão criados, compostos ou representados. Uma entidade que representa o padrão de criação usará os princípios de OO como herança para variar os tipos de classe que ela criará[Gamma 1994].

Existem dois temas principais quando se fala em padrões de criação, no primeiro temos que eles encapsulam os conhecimentos referentes a que classe, concretamente, o sistema utiliza, e no segundo temos que eles escondem como, as lógicas e regras de negocio utilizadas, as instancias desses objetos são criadas e organizadas de forma a serem utilizadas. De forma geral para o sistema, tudo que é conhecido é apenas a interface definida, o chamado contrato. Conseqüentemente os padrões de criação fornecem uma maior flexibilidade com relação ao *o que* é criado, *quem* o cria, *como* este é retornado e *quando.*

Dessa forma os objetos retornados e utilizados pelo o sistema serão compostos de valores e comportamentos que poderão ser, facilmente, alterados em qualquer momento do desenvolvimento.

Alguns padrões de Criação do GoF:

Abstract Factory: Disponibiliza uma interface para criação de uma família de objetos, relacionados ou dependentes, sem ser necessária uma especificação concreta, ou seja, não é necessário informar qual classe precisa ser criada.

Uma hierarquia que encapsula diversas possibilidades de plataformas, ou formatos, e/ou um conjunto possível de produtos.

Builder: Separa a construção de um objeto complexo da sua representação, da sua classe definida, de modo que a mesma construção possa ser usada por diferentes representações.

Possibilita a criação de entidades complexas através da criação de variados tipos que terminarão por representar tal entidade.

Factory Method: Define um mecanismo para criar um objeto, um construtor virtual, deixando que a implementação saiba e decida, qual classe será utilizada para instanciar o mesmo. Normalmente consiste em um método estático de uma classe responsável por retornar um novo objeto dessa classe.

Difere-se principalmente da Abstract Factory por retornar apenas um tipo de objeto, enquanto que a Abstract está preparada para retornar uma família de objetos.

Prototype: Define os tipos de objetos a serem criados usando uma instância padrão, cria os novos objetos copiando o protótipo. Trabalho junto com uma determinada instancia de uma classe para criar instâncias futuras.

Singleton: Garante que apenas uma instância de uma determinada classe esteja disponível no sistema provê uma interface de acesso único para esta. Encapsula uma estrutura de inicialização *Just-in-time,* apenas na hora, ou inicialização no primeiro uso.

1.2.1.2 Padrões de Estrutura.

O foco dos Padrões de Estrutura, ou Estruturais, é definir como as classes e objetos são combinados de forma a criar sistemas maiores e mais complexos. Um exemplo simples é como combinar duas ou mais classes que já sejam produtos de uma herança em uma terceira. O resultado dessa estrutura é uma classe que combina as propriedades das duas classes pais originais, conceitualmente esse padrão é particularmente útil para fazer com que classes independentes possam trabalhar juntas[Gamma 1994].

Indo além dos padrões de projeto, trabalhar com estruturas significa prover novas funcionalidades aproveitando-se os comportamentos de diferentes classes já criadas.

Alguns padrões de Estrutura do GoF:

Adapter: Converte a interface de uma determinada classe em outra. Um Adapter transforma a classe alvo em uma forma que o cliente que a utilizará possa entender. Permite que diferentes classes possam interagir através do uso das novas interfaces criadas pelo adapter.

Bridge: Separa uma abstração da sua implementação de modo que ambos possam trabalhar de forma independente.

Composite: Compõe objetos em estruturas de árvore para representar uma hierarquia do tipo todo-parte. Composite permite que os clientes, utilizadores das entidades, utilizem os objetos individuais ou combinações deles da mesma forma.

Decorator: Atribui responsabilidades dinamicamente a um objeto. Decorators fornecem uma alternativa flexível a uma sub-classe para que essa consiga ter mais funcionalidades.

Facade: Fornece uma interface única, e unificada, para um conjunto de interfaces em um subsistema. Facade Define uma interface de alto nível que permite que o sistema seja mais fácil de ser utilizado, por fornecer apenas um ponto único de acesso.

1.2.1.3 Padrões de Comportamento.

O foco dos padrões de comportamento gira em torno dos algoritmos e atribuições de responsabilidades e comportamentos entre objetos, descrevem não apenas um padrão de objetos ou classes, mas também um padrão para comunicação entre eles. Esses Padrões caracterizam complexos fluxos de controle que normalmente são complicados de ser acompanhado em tempo de compilação. Eles buscam tirar o foco do desenvolvedor do

fluxo de controle para que ele possa se concentrar apenas na forma como os objetos estão interligados.

Alguns Padrões de Comportamento do GoF:

Iterator: Provê um mecanismo para acessar os elementos dentro de um conjunto de objetos, sistema, sem expor suas representações.

Observer: Define uma dependência 1-N (um para N) entre objetos, de forma que quando determinada ação ocorra em um objeto este tenha uma maneira de notificar, os N objetos, do evento ocorrido.

Encapsula o núcleo dos componentes em uma abstração e varia os componentes de modo a possibilitar diferentes notificações.

Command: Encapsula as requisições a um objeto permitindo, dessa forma, que o desenvolvedor possa parametrizar os clientes com diferentes tipos de requisição.

Memento: Sem violar o encapsulamento, captura e retorna ao exterior, o estado interno de um objeto, de modo que o objeto em questão possa ser retornado ao seu atual posteriormente.

Strategy: Define uma família de algoritmos, encapsula cada um deles, e faz com que eles possam ser intercambiáveis. Strategy permite que os algoritmos possam variar independentemente do cliente que o usa.

1.2.1.4 Abordagens a Padrões de Projeto

Existem diversos estudos focados em padrões de projeto. Por um lado temos estudos como o de Yoder[Yoder 1998] que foca em padrões de segurança mas afirma que padrões de projeto auxiliam na definição de arquiteturas de softwares, por outro temos estudos como os apresentados por Khomh[Khomh 2008][Haley 2004] onde ele mostra que padrões de projeto nem sempre se apresentam como uma boa solução do ponto de vista da qualidade de softwares por impactar negativamente em uma serie de critérios de qualidade.

Diferente desses, trabalhos como o de Sandhu tentam mostrar que Design Patterns, ou Padrões de Projeto, podem acelerar o processo de desenvolvimento de software oferecendo passos, pré-testados varias vezes, a serem seguido. Design Patterns mostram sua eficiência em todas as fases desde o desenvolvimento até a manutenção e podem ser analisados pela sua qualidade, fator esse que é um dos aspectos mais relevantes na avaliação de qualquer estrutura de software. Sandhu[28] faz uma analise de um conjunto de métricas que podem ser aplicáveis e válidas para todos os padrões. Sua proposta foca principalmente em analisar a quantidade de padrões que foram adotados e utilizados em um determinado projeto, para que através da detecção de um conjunto desses padrões o seu framework proposto possa indicar um valor relacionado a uma de suas métricas que por sua vez são relacionadas à qualidade do sistema de maneira geral.

Já Outros trabalhos como o de Mario Luca e Giuseppe mostram que Padrões de projeto apresentam uma serie de problemas relacionados à qualidade do código gerado,

afetando de forma negativa métricas como modularidade. Nesse trabalho é apresentada uma abordagem onde podemos fazer ver re-implementações de padrões de projeto através de orientação a aspectos de uma forma bastante efetiva de modo a diminuir tais impactos negativos. É-nos mostrado que orientação a aspectos permite criar diferentes soluções para os problemas citados mesmo que estas gerem novos problemas como coesão e acoplamento, inerentes a próprio Orientação a Aspectos.

Rudzki[Lutz 2000] foca principalmente em como o uso de design patterns na montagem de interfaces remotas influencia na performance de aplicações distribuídas. Os padrões estudados por Rudzki são os considerados como boa opções para design de aplicações.

Para Rudzki performance é apenas um entre muitos atributos de qualidade que podem ser utilizados para descrever as propriedades de uma aplicação, mas em muitos casos tais propriedades são colocadas em uma posição alta de listas de prioridades relacionadas a requisitos funcionais. Dessa forma qualquer medida que possa ser tomada durante o processo de desenvolvimento que auxilie na melhora da performance é benéfico uma vez que seu trabalho concluiu que decisões relacionadas ao uso de um determinado design pattern em um sistema distribuído impacta na performance do sistema como um todo.

Analisados de forma positiva ou negativa, é fácil perceber que existem um série de estudos focados em padrões de projeto, sejam analisando seu impacto ou aprovação, tais estudos já fazem parte do dia a dia dos desenvolvedores e equipes de software. Indo alem, já do ponto de vista de requisitos não funcionais, podemos verificar que performance da mesma forma que segurança aparece entre os requisitos não funcionais relativos a qualidade e como aqueles onde se perceber a maior negligência por parte de membros das equipes.

Em virtude dessas considerações nosso próximo estudo será conduzido com relação a requisitos de segurança, focando principalmente em seu entendimento.

1.2.2. Requisitos de Segurança

A engenharia de requisitos dentro de um negócio, sistemas, aplicações e componentes é muito mais do que apenas documentar os requisitos funcionais destes. Mesmo que, grande parte dos analistas se dedique a elicitar alguns requisitos de qualidade como: interoperabilidade, disponibilidade, performance, portabilidade e usabilidade, muitos deles ainda pecam no que diz respeito a analisar questões relacionadas a Segurança.

Infelizmente, documentar requisitos de segurança específicos é difícil. Estes tendem a se mostrar como de alto impacto para muitos requisitos funcionais. E mais, requisitos de segurança são, normalmente, expressados de forma a documentar os termos de como alcançar segurança a não na forma do problema que precisa ser resolvido[Haley 2004].

A maior parte dos analistas de requisitos não tem conhecimentos na área de Segurança, os poucos que receberam algum tipo de treinamento tiveram apenas uma visão geral sobre alguns mecanismos de segurança como senhas e criptografia ao invés de conhecer reais requisitos nessa área[Yoder 1996][Firesmith 2003][Firesmith 2004]

Requisitos de segurança tratam de como os bens de um sistema devem ser protegidos contra qualquer tipo de mal[Haley 2004][Haley 2006]. Um bem é algo no contexto do sistema, tangível ou não, que deve ser protegido[ISSO IEC]. Um mal ou ameaça da qual um sistema precisa ser protegido, é uma possível vulnerabilidade que pode atingir um bem. Uma vulnerabilidade é uma fraqueza de um sistema que um ataque tende a explorar. Requisitos de segurança são restrições nos requisitos funcionais com o intuito de reduzir o escopo das vulnerabilidades[Haley 2004].

Donald Firesmith consolida de forma bastante simples os requisitos de Segurança encontrados mais comumente, são eles[Firesmith 2003]:

•**Requisitos de Identificação:** Definem o grau e mecanismos que uma entidade utiliza para conhecer, ou reconhecer, entidades externas a ela como por ex: usuários, aplicações externas ou serviços, antes que estes possam interagir. Exemplos de Requisitos de Identificação:

- A aplicação deve identificar todas as suas aplicações clientes antes de permitir que elas tenham acesso a suas funcionalidades.
- A aplicação deve identificar todos os usuários antes de permitir que eles tenham acesso a suas funcionalidades.

•**Requisitos de Autenticação:** Definem o grau e mecanismo que uma entidade verifica, confirma ou nega, a identidade apresentada pelos externos, usuários, aplicações externas ou serviços, antes que estes possam interagir. Exemplos de Requisitos de Autenticação:

- A aplicação deve verificar a identidade de todos os seus usuários antes de permitir que eles tenham acesso a suas funcionalidades.
- A aplicação deve verificar a identidade de todos os seus usuários antes que estes possam alterar algum tipo de dado do sistema.

•**Requisitos de Autorização:** Definem o grau de acesso e privilégios que determinada(s) entidade(s) ou perfis receberá após ser autenticada e validada por alguma parte do sistema. Exemplos de Requisitos de Autorização:

- A aplicação deve permitir que cada usuário tenha acesso a todos os seus dados pessoais.
- A aplicação não poderá permitir que usuários tenham acesso às informações dos demais usuários.

•**Requisitos de Imunidade:** Define o grau em que uma entidade protege a si mesma de invasões, infecções ou alterações, por parte de programas maliciosos, por ex: Vírus, trojans, worms e scripts maliciosos.Exemplos de Requisitos de Imunidade:

- A aplicação deve conseguir se desinfetar de todo e qualquer arquivo, que seja detectado como danosos, se possível.

- A aplicação deve notificar o administrador de segurança e usuário logado, em caso de detecção de algum programa danoso, durante um processo de scan.

•**Requisitos de Integridade:** Define o grau no qual uma comunicação, ou dado, hardware e software, se protegem de tentativas de corrupção por parte de terceiros. Exemplos de Requisitos de Integridade:

- A aplicação deve prevenir a corrupção, não autorizada, de dados, mensagens e valores e qualquer outra entidade enviada pelo usuário.
- A aplicação deve garantir que os dados armazenados nela não estejam corrompidos.

•**Requisitos de Detecção de intrusão:** Define o grau no qual uma tentativa de acesso ou modificação não autorizado é detectado, registrado e notificado, pelo sistema. Exemplos de Requisitos de Detecção de intrusão:

- A aplicação deve detectar e registrar todas as tentativas de acesso que falharem nos requisitos identificação, autorização ou autenticação.
- A Aplicação deve notificar os responsáveis imediatamente sempre que algum perfil sem a correta permissão tente acessar uma área restrita mais de uma vez.

•**Requisitos de Não – Repudiação:** Define o grau no qual uma parte de um determinado sistema impede uma das partes das interações tomadas dentro, ou pelo sistema, por ex: uma troca de mensagens, neguem seu envolvimento em determinado momento.Exemplos de Requisitos de Não – Repudiação:

- A aplicação deve ser capaz de armazenar dados sobre as transações feitas por um usuário de modo a conseguir identificar com precisão qual usuário efetuou qual transação.
- A aplicação deve ser capaz de armazenar dados sobre as ações feitas por cada usuário de modo a conseguir identificar com precisão qual usuário efetuou qual ação.

•**Requisitos de Privacidade:** Também conhecido como Confidencialidade. Define o grau no qual dados importantes e comunicações são mantidos privados de acesso por parte de indivíduos ou programas. Exemplos de Requisitos de Privacidade:

- A aplicação deve garantir que usuários não possam acessar dados confidenciais de outros usuários.
- A aplicação deve garantir que toda e qualquer comunicação feita pelos usuários não poderá ser entendida em caso de captura.

•**Requisitos de Auditoria de Segurança:** Define o grau em que a equipe responsável pela segurança tem permissão para verificar o status e uso dos mecanismos de segurança

através da analise de eventos relacionados a estes. Exemplos de Requisitos de Auditoria de Segurança:

- A aplicação deve ser capaz de detectar e armazenar todas as transações feitas pelo usuário
- A aplicação deve prover um mecanismo através do qual o pessoal de administração consiga rastrear as ações de cada usuário dentro do sistema.

- **Requisitos de Tolerância a Falhas:** Define o grau em que uma entidade continua a executar sua missão, provendo os recursos essências a seus usuários mesmo na presença de algum tipo de ataque. Exemplos de Requisitos de Tolerância:

- A aplicação só pode apresentar um único ponto de falha.
- A aplicação não pode ter seu sistema de auditoria fora do ar.

- **Requisitos de Proteção Física:** Define o grau em que uma entidade se protege fisicamente de outra entidade. Exemplos de Requisitos de Proteção Física:

- Os dados armazenados em hardware devem ser protegidos contra danos físicos, destruição, roubo ou troca não autorizada.
- Aqueles que manuseiam os dados devem ser protegidos contra danos, morte e seqüestro.

- **Requisitos de Manutenção de Segurança de Sistemas:** Define o grau em que o sistema deve manter suas definições de segurança mesmo após modificações e atualizações.Exemplos de Manutenção de Segurança de Sistemas:

- A aplicação não deve violar seus requisitos de segurança após upgrade de dados, hardware ou componentes.
- A aplicação não deve violar seus requisitos de segurança após uma troca de dados, hardware ou componentes.

1.3. Relacionando Requisitos de Segurança e Padrões de Projeto

Nesse capítulo faremos uma divisão e classificação dos tipos de requisitos mencionados anteriormente e faremos a proposta do seu relacionamento com padrões de projeto, a fim de tentar garantir uma abordagem relacionada à implementação desses requisitos utilizando as especificações feitas pelo GoF[Gamma, 1996] de modo a tornar mais factível a implementação de requisitos de segurança.

A adoção desses padrões, seguindo a proposta neste trabalho, auxilia a construção do sistema garantindo que definições relacionadas aos requisitos de segurança possam ser mais bem estruturadas, alem de proporcionar uma maior facilidade relacionada às implementações das políticas de segurança mesmo que essas sejam definidas apenas em um segundo momento.

1.3.1 Divisão e Classificação

Uma vez definidos os tipos de requisitos de segurança e analisadas suas definições, iremos classificá-los de acordo seus propósitos, agrupando aqueles que mostrarem mais semelhança.

Inicialmente, tomaremos os requisitos de: Identificação, Autenticação, Autorização, Não Repudiação e Privacidade. Ao analisar esses requisitos podemos observar que todos tratam do mesmo assunto, *identificação de um ator*, seja ele usuário, sistema ou outra entidade que interage com o sistema em questão. Todos tratam da identidade que este ator possui. Respectivamente temos a Identificação do ator, a prova da Identificação, as permissões da identidade, a confirmação das ações da entidade e por fim os segredos ou sigilo da identidade. Todos esses requisitos giram em torno da criação da Identidade do individuo desde os primeiros passos, ou no caso de sistemas, telas, interações, casos de uso ou outros.

É possível visualizar também que, uma vez falado em criação de entidades relacionadas identidade, a arquitetura do sistema sofrerá impactos para que seja possível a adoção, e utilização, das identidades criadas. Indo além, podemos verificar que uma vez adaptada a arquitetura para a necessidade em questão, relacionada à criação da identidade, também será necessário adaptá-la para fazer uso dessa identidade.

Continuando a analise dos requisitos, podemos separar os requisitos de *Imunidade* e *Integridade*, como sendo dois requisitos que atingem diretamente a estrutura do sistema. Do ponto de vista deste, os requisitos de *Imunidade* visão garantir que o sistema esteja imune a contaminações por partes dos atores e os requisitos de *Integridade* visão garantir que a estrutura do sistema proveja um mecanismo integro para a comunicação entre esses atores. Dessa forma ambos os requisitos tem uma relação direta com a estrutura do sistema uma vez que será necessário alterar a estrutura do sistema de modo a adotar soluções para esses requisitos.

Seguindo, temos os requisitos de *Detecção de Intrusão, Auditoria de Segurança* e *Tolerância a Falhas*, que tratam das questões relacionadas às ações tomadas junto ao sistema. No primeiro caso, *Detecção*, temos um requisito que trabalha de forma preventiva, que visava disponibilizar mecanismo para detecção e notificação em caso de acesso indevido; já a auditoria vem como um mecanismo para trabalhar questões de forma mais reativa, ou seja atitudes que podem ser tomadas a partir da comprovação e constatação de ações, um requisito de auditoria deve contemplar o registro de ações como também mecanismos para futuras consultas[Schumacher, 2006], diferente de *Tolerância a falhas* que além de reativo, define qual o comportamento o sistema terá em caso de falha, também é preventivo ao garantir que falhas em entidades do sistema não comprometam o restante do sistema. Por tanto, os requisitos mencionados trabalham a questão dos comportamentos tomados dentro do sistema.

Por fim analisando os requisitos de Manutenção de Segurança do Sistema e de Proteção Física temos, nesse, um requisito que trata mais da questão física, como o próprio nome remete, aonde a preocupação vai além do âmbito software, por tanto fora

do nosso escopo de analise. Já o requisito de Manutenção tem um comportamento horizontal em relação aos demais requisitos, visto que esse trata da manutenção das demais necessidades do sistema relacionadas à segurança, indiretamente ele trata dos requisitos responsáveis por tais necessidades. Aparece por tanto um requisito não considerável do ponto de vista de software e um que é a soma de demais requisitos.

Organizando-os de acordo com suas características temos:

1-Requisitos de Identificação, Autenticação, Autorização, Não-Repudiação e Privacidade sendo relacionados com criação.

2-Requisitos de Imunidade e Integridade relacionados com a estrutura do sistema.

3-Requisitos de Detecção de Intrusão, Auditoria e Tolerância a Falhas relacionados com comportamentos dos atores no sistema.

4-Requisitos de Manutenção de Segurança de Sistemas relacionado com os demais requisitos.

Sob essa óptica e utilizando umas das classificações do GoF, podemos separa os requisitos de acordo com seus propósitos. A partir das características apresentadas, vamos separá-los em 3 grupos de acordo com esse critério propósitos, são eles, Criação, Estrutural e Comportamental. A tabela 1 mostra essa divisão:

Como mencionado anteriormente, Os requisitos de Proteção física, por tratarem de questões físicas relacionadas ao ambiente físico do sistema não são contemplados dentro dessa estrutura.

Propósito		
Criação	**Estrutural**	**Comportamental**
Requisitos de Identificação	Requisitos de Imunidade	Requisitos de Detecção de intrusão
Requisitos de Autenticação	Requisitos de Integridade	Requisitos de Auditoria de Segurança
Requisitos de Autorização		Requisitos de Tolerância a Falhas.
Requisitos de Não–Repudiação		
Requisitos de Privacidade		
Requisitos de Manutenção de Segurança de Sistemas		

Tabela 1: Divisão de requisitos de acordo com os tipos de seus propósitos.

1.3.2 Relação com Padrões de Projeto

Conhecendo agora a divisão da sessão anterior, o próximo passo é tentar tratar cada um desses requisitos, de acordo com seus propósitos, de forma a tentar garantir uma relação com padrões de projeto.

É importante lembrar que diferente dos *Security Patterns*[Schumacher, 2006], que se apresentam em sua maioria como Padrões Arquiteturais e da abordagem proposta por Weis, Michael[Weiss 2008] estamos abordando o uso de Padrões de Projeto na construção de arquiteturas com o intuito de tornar a adoção dos requisitos de segurança, desses sistemas, mais simples e próxima aos desenvolvedores, que, como mencionado anteriormente, tem pouco ou nenhum conhecimento do ponto de vista de segurança de software, alem disso podemos também diminuir os impactos de implementações tardias das políticas de segurança uma vez que muitas vezes estas não estão claras no inicio da criação dos sistemas.

Como vimos, o GoF[Gamma, 1996] classifica seu padrões por dois critérios, sendo o critério de propósito responsável por classificar os padrões de acordo com o que o padrão faz. Dessa forma existem padrões responsáveis pela criação de entidades, estrutura do sistema e comportamento das entidades, respectivamente: Padrões de Criação, Estrutura e Comportamento. Sendo assim é possível ver um relacionamento direcionado entre tipos de requisitos e padrões. Figura. 2

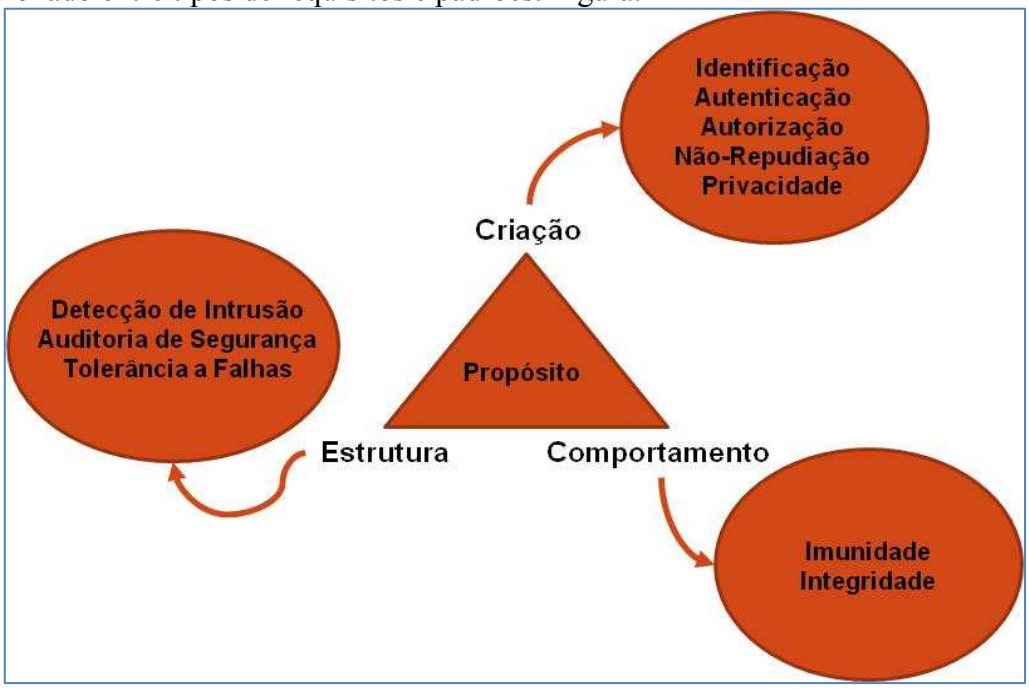


Figura 2: Divisão dos requisitos.

Uma vez divididos os tipos de requisitos e os relacionando com padrões de projeto, resta agora entender como utilizá-los de modo a auxiliar a criação das arquiteturas a fim de diminuir os impactos.

1.3.3. Uso

Por fim, uma vez entendida a relação entre os requisitos e seus respectivos padrões de projeto iremos fazer uso dessa relação, para trabalhar a situação mencionada no capítulo 3. Vamos propor uma abordagem utilizando essencialmente OO e padrões de projeto para tentar montar uma solução de software que esteja preparada para receber implementações de segurança mesmo que estas não tenham sido completamente definidas durante etapas de iniciais do projeto. Respaldado na afirmação de Yoder que mostra que padrões de projeto auxiliam na construção de arquiteturas[Yoder, 1996] utilizaremos a relação criada na sessão anterior para, através da adoção dos padrões, trabalhar problemas de segurança. Nesse situação é importante que foquemos no uso dos padrões de forma correta e não no retorno ou comportamentos do padrão, por tanto sempre recomendaremos o uso de implementações *stubs* dos padrões em questão, para que esses possam ser adicionados ao código e criando o mínimo impacto nas execuções iniciais do sistema e diminuindo impactos de modificações futuras.

Importante salientar que estamos sugerindo uma abordagem em código para um problema conhecido arquitetural, dessa forma estamos trabalhando com as soluções apresentadas pelo GoF para resolver problemas relacionados a requisitos de segurança de software e não hardware.

Para os relacionamentos entre requisitos e padrões mostrados anteriormente, teremos situações relacionadas à Criação, Estrutura e Comportamento do sistema, tais podem ser abordados como mostrado nas sessões seguintes.

Por questões meramente de visualização utilizaremos exemplos usando a linguagem Java, desde já deixamos claro que não será utilizada nenhuma API específica na nossa abordagem, mesmo que em um dos estudos de caso apresentados utilize tecnologias especificas dessa plataforma.

1.3.4 Criação.

Os requisitos relacionados à criação devem ser tratados através da adoção de algum dos padrões criacionais do GoF.

Requisitos de Identificação, Autenticação e Autorização tratam da criação da(s) entidade(s) responsável(is), respectivamente, pela identificação do individuo ou sistema, da validação do mesmo e das permissões que este possui. Estes por si só utilizarão um padrão de criação que será responsável por criar as tais entidades, e se for o caso, com seus métodos vazios caso maiores informações relacionadas ao requisito não sejam informadas.

Tomemos como exemplo um *Factory Method*[Gamma 1996] ou *Abstract Factory*[Gamma 1994] na Figura 3.

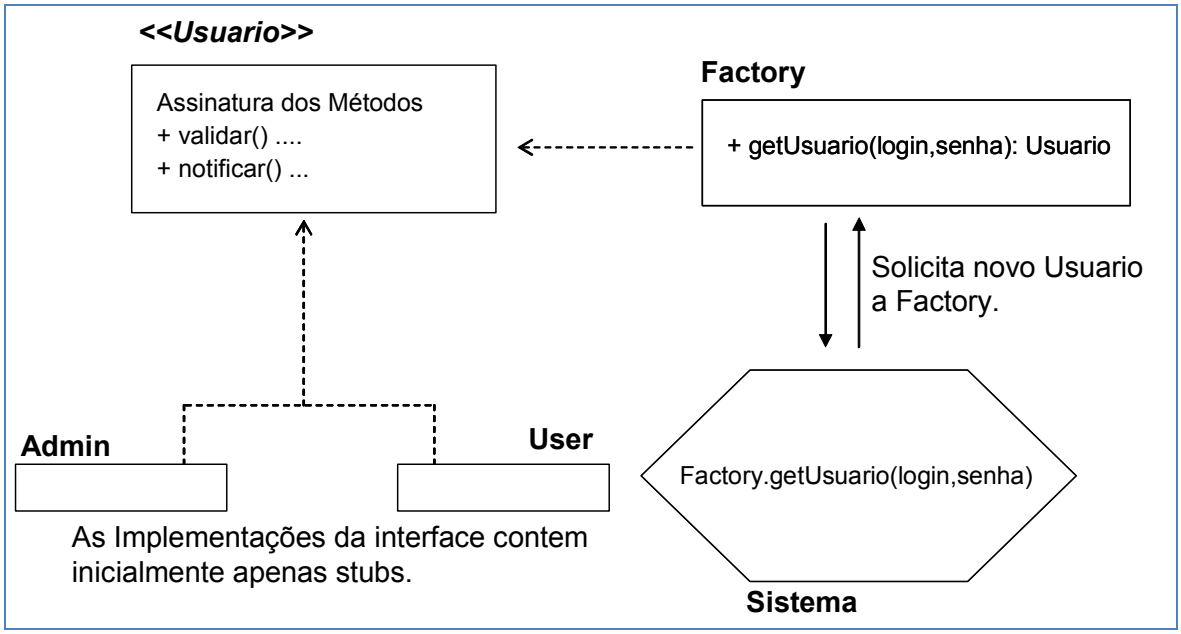


Figura 3: Esquema de criação da entidade Usuário utilizando Factory

A Factory em questão contemplará:

- A definição de uma interface
- A criação de uma classe para retornar as instâncias das implementações dessa interface.
- A criação de implementações da interface definida.

A *Factory* em questão, deve estar apta a retornar as implementações definidas sempre que for requisitada pelo sistema, no exemplo a cima adotamos 2 entidades retornáveis pela *Factory*, são elas *Admin* e *User*, que representam usuários do sistema do tipo Administrador e Usuário, ambos com permissões a serem definidas. A interface comum a essas duas implementações é chamada *Usuário* e possui métodos relacionados à validação das permissões dessa entidade.

O sistema deverá fazer uso dessa, ou de demais *Factorys* sempre que for necessária a criação das entidades relacionadas aos atores do sistema ou sempre que se julgar necessária a criação de uma nova entidade relacionada a algum requisito de Criação. Nesse momento o *Factory* retornará uma instância do tipo *Usuario* que, no exemplo, poderá ser tanto *Admin* quanto *User* e estes por sua vez contem a implementação da interface *Usuário*, tendo seus métodos implementados na forma de *stubs*, em outras palavras, os métodos estarão vazios ou retornando valores positivos, por exemplo *true* para valores *boleanos*. Tomemos a Figura 2 para ilustrar a implementação *stub* da classe *Admin*.

```
public class Admin implements Usuario {

        public boolean validar() {
            // TODO: Modificar esse bloco, acrecentando
            // a correta validação do Usuario
            return true;
        }

        public boolean temPermissao(int id) {
            // TODO: Modificar esse bloco, acrecentando
            // a correta verificação das permissões do Usuario
            return true;
        }

}
```

Figura 4: Exemplo de Stub.

Na figura 4, podemos ver os métodos *validar()* e *temPermissao()* retornando o valor *true* para qualquer caso. Dessa forma mesmo que o foco do desenvolvedor seja, inicialmente, validar algum requisito relacionado a algum requisito funcional do sistema, ao utilizar essa simples estrutura podemos garantir que mesmo o foco do desenvolvedor sendo outro, essa adoção não trará impactos nesse momento inicial e os reduzirá em caso de mudanças no futuro quando as reais especificações e validações relacionadas a políticas de segurança serão definidas.

Dentro ainda dos Requisitos ligados a Criação, temos ainda os requisitos de Não–Repudiação e de Privacidade que devem utilizar os objetos criados anteriormente pela *Factory* em questão. Dessa forma temos como garantir que, para o requisito de Não-Repudiação, as atividades, ou ações, executadas pelos usuários do sistema não possam ser negadas desde que sejam associadas às identidades criadas e disponibilizadas pela *Factory*, ou seja, ao se aplicar também critérios de auditoria temos um mecanismo de identificação que será utilizado para criar relação entre Ator-Ação onde o ator é, inicialmente, um *stub* a ser modificado posteriormente e a ação será também um *stub*, como veremos a seguir.

Para garantir e adotar o requisito de Privacidade teremos um mecanismo onde utilizaremos as entidades criadas, associadas as suas permissões, utilizando-as de modo a garantir que apenas o usuário com a correta permissão para tal, possa interagir com alguma informação, dado ou até mesmo área do sistema.

É importante mostrar que estamos trabalhando inicialmente com o uso de uma Factory, mas apenas como uma sugestão, no estudo de caso veremos que um padrão relacionado ao propósito de comportamento(*command*), juntamente com um criacional(*singleton*) serão utilizados para implementar requisitos de Autenticação e Autorização.

1.3.5 Estrutural.

Os requisitos relacionados à estrutura do sistema serão classificados nessa categoria. São eles: Requisitos de Imunidade e Requisitos de Integridade.

Mais do que tratar de como a estrutura do sistema será montada, ou como classes poderão ser adaptadas para outros usos[Gamma, 1996], essa categoria também se preocupa em montar o sistema de forma a facilitar o relacionamento entre as entidades através do uso de informações presentes no sistema como base para decisão de ações a serem tomadas.

Tomemos como exemplo o padrão *Adapter,* esse padrão tem como principal característica permitir modificações posteriores nas entidades do sistema, garantindo o mínimo de mudanças estruturais. Ele provê um mecanismo para transformar uma entidade em outra através da adaptação das chamadas de seus métodos[Gamma 1996].

No nosso exemplo, os objetos que transitarem pelo sistema serão fruto das criações de identidade e permissões dos *Usuários*. Uma vez criado os objetos, poderemos alterá-los para retornar valores ou ter comportamentos diferentes através da adoção do *Adpater*. Nesse cenário a interface definida anteriormente não será modificada, mas sim utilizada para checar as permissões e validações, tais comportamentos por sua vez serão utilizados pelo *Adapter* que implementará a interface e internamente repassará os comportamentos para as entidades adequadas, em outras palavras, o adaptador será o ponto de entrada para diferentes verificações a partir da interface e objetos conhecidos. Esses novos objetos deverão ser utilizados para garantir os requisitos de Imunidade e Integridade.

No exemplo iniciado na sessão anterior, queremos que o requisito de Imunidade garanta que apenas usuários, ou entidades, com permissão adequada possam ter acesso ao ambiente do sistema, ou permissão para escrita, leitura, ou qualquer outro tipo de ação, isso será alcançado através da checagem do objeto que representa o usuário e suas permissões. Futuramente ao se modificar as entidades para que essas possam executar diferentes tipos de validações e checagens, ou até mesmo executar diferentes procedimentos quando seus métodos conhecidos forem chamados, utilizaremos um Adaptador para que as modificações imposta a entidade sejam mínimas e não alterem as assinaturas dos métodos, dessa forma temos que o adaptador ficará responsável por receber as chamadas do sistema e de encaminhá-las a entidade correta. Figura 5 ilustra a idéia do Adaptador.

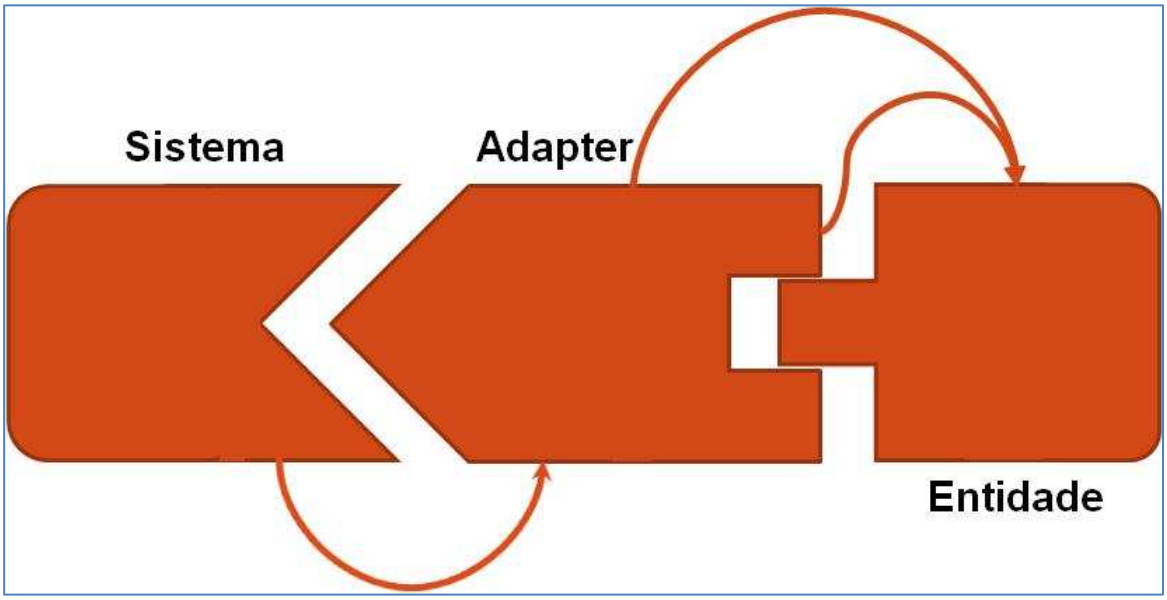


Figura 5: Ilustração do adaptador.

Sendo assim, será criada uma nova estrutura com as mesmas características da entidade adaptada.

O novo objeto deve ser checado sempre que acessos a algum dado ou hardware sejam efetuados, para que se possa garantir também o requisito de Integridade. Nesse cenário trabalharemos com uma checagem de permissão para que apenas entidades autorizadas possam executar determinadas alterações. Figura 6.

```
if(user.temPermissao(Usuario.PERMISSAO_ESCRITA)) {
                    //Executar restante do código
}
```

Figura 6: Checagens do Usuário.

Ainda nesse cenário, pode-se estender esse tipo de abordagem para acrescentar às funcionalidades de verificação, algoritmos para garantir a integridade também do conteúdo dos dados acessados. Nessa, poderemos fazer uso do adaptador para modificar posteriormente tais verificações modificando os algoritmos para melhor atender as necessidades do sistema.

1.3.6 Comportamental.

Os requisitos relacionados aos comportamentos do sistema são classificados nessa categoria. Requisitos de Detecção de intrusão, Auditoria de Segurança e Tolerância a Falhas tratam da notificação do sistema em caso de determinados eventos ocorrerem, bem como da atitude tomada pelo sistema quando tais eventos ocorrem. No primeiro

caso eventos de falha relacionados a acessos indevidos, tanto a sua detecção quanto ação quando dos acontecimentos. No segundo há uma gama maior de possibilidades que devem ser primeiro detectados e em seguida armazenados para futura consulta. Por fim temos ações tomadas para a prevenção de falhas junto ao sistema.

Para esse tipo de situação propõe-se a adoção dos padrões relacionados ao Comportamento. Por ex, *Observer*.

Teremos uma entidade responsável, o *Observer*, por receber mensagens, relacionadas a determinados eventos. Para eventos relacionados à intrusão deve-se avisar ao *Observer* para que esse tome alguma ação desde notificar os responsáveis até mesmo negar o acesso, para eventos relacionados à auditoria devemos notificar o responsável para que esse armazene os eventos enviados de modo a possibilitar uma futura consulta e por fim em caso de eventos provindos da detecção de falhas avisarem ao *Observer* para que esse tome alguma atitude a fim de tornar transparente ao usuário do sistema que tal falha aconteceu. A Figura 7 mostra uma abordagem geral para o caso do *Observer*.

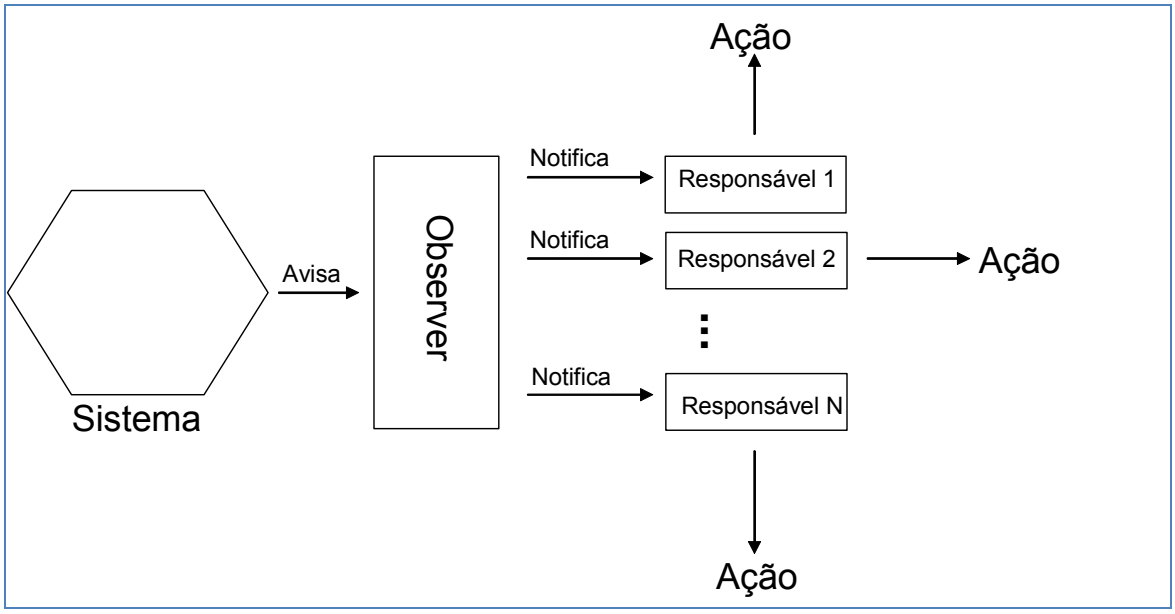


Figura 7: Notificações ao Observer

A partir dessas notificações o *Observer* deverá notificar todas as entidades nele registradas. Todas as entidades que desejarem ser notificadas em caso de algum tipo ação relacionadas à Auditoria, Detecção de Intrusão ou Tolerância a Falhas deverão implementar uma interface especifica e se registrar no *Observer* correto.

Uma vez registradas as entidades junto ao *Observer* e este recebendo a notificação de alguma ação as entidades deverão ser notificadas a fim de efetuar alguma ação sejam elas bloquear acesso, restringir uso, reiniciar sistema ou mesmo apenas notificar os responsáveis. Inicialmente tal implementação da interface deverá ser vazia de modo a que essencialmente nenhuma ação seja tomada, quando as políticas relacionadas a tais requisitos forem determinadas a estrutura para tomada de ações já estará montada.

1.3.7 Requisito de Manutenção de Segurança de Sistemas.

Por fim tratamos do requisito mais abrangente. Pela definição de Firesmith[Firesmith 2003][Firesmith 2004], vemos que esse requisito cuida do sistema após a instalação de uma nova versão, garantindo que antigas definições de segurança não sejam perdidas. Através da padronização do código do sistema pelo uso dos padrões referentes à aos requisitos de Criação, Comportamento e Estrutura, teremos a possibilidade de uma atualização mais fácil, inicialmente os *stubs* serão criados e aplicados, se em algum momento modificações relacionadas ao escopo de segurança forem necessárias é possível fazê-las apenas preenchendo os métodos vazios. Tal procedimento tende, nesse momento, a ser mais fácil e menos custoso uma vez que os locais passíveis de alteração já são conhecidos, bem como os impactos de suas chamadas.

Espera-se que as atualizações dos requisitos de segurança, sejam efetuadas de forma menos danosa uma vez que agora adotam um padronização em seu código através da utilização dos padrões na definição e construção de sua arquitetura.

1.3.8 Consolidação

A Tabela 2 mostra de forma consolidada a relação criada entre requisitos, padrões sugeridos e sua utilização.

	Requisitos	Padrões	Uso
Criação	Identificação Autorização Autenticação Não Repudiação Privacidade	Factory Method Or Abstract Factory	Criar interfaces e implementações retornáveis pelo Factory sempre que alguma entidade responsável por identificação for criada.
Estrutura	Imunidade Integridade	Adapter	Faça com que as diversas áreas do sistema chequem as identidades criadas anteriormente para validar permisoes. Utilize o Adapter para mudar essas entidades.
Comportamento	Detecção de Intrusão Auditoria Tolerância a Falhas	Observer	Faça com que os responsáveis pelas ações registrem-se junto ao Observer para que este notifique-as em casos específicos. Faça com que essas entidades estejam vázias.

Manutenção de requisitos de segurança

Tabela 2: Resumo das Atribuições.

A intenção desse trabalho foi mostrar que existe uma maneira fácil e ágil para se implementar requisitos de segurança. Como foi levantado, existe uma grande negligência no que diz respeito a esse aspecto em particular, muitas vez por falta de detalhes e outras por desconhecimento, espera-se que através da implementação desses padrões as futuras, implementações muitas vezes na fase final do desenvolvimento, possam ser feita garantindo um mínimo de qualidade e aceitação para o produto sem um grande impacto de cronograma. Para tal, inicialmente analisamos a classificação proposta pelo GoF, em seguida analisamos e entendemos os tipos de requisitos de segurança, posteriormente classificamos tais requisitos com base em uma relação com padrões de projeto e por fim mostramos o como utilizá-los sem prejudicar o projeto, criando uma estrutura baseada nos requisitos deste para garantir segurança da informação dentro do contexto de software.

Os pontos abordados nesse trabalho seguem apenas como uma sugestão inicial no que diz respeito a que padrões utilizarem para cada requisito. Cabe aos arquitetos, analistas, desenvolvedores avaliarem melhores os Padrões de Projetos apresentado pelo GoF, adaptando-os de forma semelhante a apresentada no trabalho, adequando dessa forma sua realidade a uma arquitetura mais apropriada, atendendo os requisitos de segurança que forem necessários.

Referências

7498-2, ISO. 1989. *Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture*.

Adikari, Jithra. 2006. "Efficient Non-Repudiation for Techno-Information Environment." *First International Conference on Industrial and Information Systems* 454-458. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4216631.

Devanbu, Premkumar T, and Stuart Stubblebine. 2000. "Software Engineering for Security: a Roadmap. "pp. 227-239 in *The future of Software Engineering*. ACM Press.

Dhillon, Gurpreet. n.d. *Principles of Information Systems Security: Texts and Cases*. 1 edition. Wiley.

Firesmith, D.G. 2004. "Analyzing and Specifying Reusable Security Requirements. "in *Eleventh International IEEE Conference on Requirements Engineering (RE'2003) Requirements for High-Availability Systems (RHAS'03) Workshop*. Citeseer

Firesmith, D.G. 2003. "Engineering security requirements." *Journal of Object Technology* 2:53–68.

Gamma, E, R Helm, R Johnson, and J Vlissides 1994. n.d. *Design Patterns: Elements of Reusable Object-Oriented*. 1st edition. Addison-Wesley Professional

Haley, Charles B, Jonathan D Moffett, Robin Laney, and Bashar Nuseibeh. 2006. "A framework for security requirements engineering. "pp. 35-42 in *SESS '06: Proceedings of the 2006 international workshop on Software engineering for secure systems*. New York, NY, USA: ACM Press http://dx.doi.org/10.1145/1137627.1137634.

Haley, Charles B, Robin C Laney, and Bashar Nuseibeh. 2004. "Deriving security requirements from crosscutting threat descriptions. "pp. 112-121 in *AOSD '04: Proceedings of the 3rd international conference on Aspect-oriented software development*. New York, NY, USA: ACM Press http://dx.doi.org/10.1145/976270.976285.

Information Technology - Security Techniques - Evaluation Criteria for IT Security. Part 1: In. Geneva Switzerland: ISO/IEC Information Technology Task Force (ITTF).

Khomh, F, and Y G Gueheneuc. 2008. "Do Design Patterns Impact Software Quality Positively?. "pp. 274-278 in *Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on*. http://dx.doi.org/10.1109/CSMR.2008.4493325.

Kunyu, Peng. 2009. "AN IDENTITY AUTHENTICATION SYSTEM BASED." *Identity*.

Lutz, R.R. 2000. "Software engineering for safety: a roadmap. "p. 213–226 in *Proceedings of the Conference on The Future of Software Engineering*. ACM http://portal.acm.org/citation.cfm?id=336512.336556.

Pawlak, Piotr, Bartosz Sakowicz, Piotr Mazur, and Andrzej Napieralski. 2009. "Social Network Application based on Google Web." *Source* 461-464.

Peiris, Hasala, Lakshan Soysa, and Rohana Palliyaguru. 2008. "Non-Repudiation Framework for E-Government Applications." *2008 4th International Conference on Information and Automation for Sustainability* 307-313. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4783950.

Samarati, Pierangela, and Sabrina De Capitani di Vimercati. 2000. "Access Control: Policies, Models, and Mechanisms. "pp. 137-196 in *FOSAD*, vol. 2171, *Lecture Notes in Computer Science*, Riccardo Focardi and Roberto Gorrieri. Springer.

Schumacher, Markus, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, and Peter Sommerlad. 2006. *Security Patterns : Integrating Security and Systems Engineering (Wiley Software Patterns Series)*. John Wiley & Sons http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0470858842.

Sindre, Guttorm, and Andreas L. Opdahl. 2004. "Eliciting security requirements with misuse cases." *Requirements Engineering* 10:34-44. http://www.springerlink.com/index/10.1007/s00766-004-0194-4.

Steiner, Peter. 1993. "On the internet nobody knows you´re a Dog." *The New Yorker* 69:61.

Wang, Zhi-Hui, Ming-Chu Li, Mao-Hua Chen, and Chin-Chen Chang. 2009. "A New Intelligent Authorization Agent Model in Grid." *2009 Ninth International Conference on Hybrid Intelligent Systems* 394-398. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5254309.

Weiss, Michael, and Haralambos Mouratidis. 2008. "Selecting Security Patterns that Fulfill Security Requirements." *2008 16th IEEE International Requirements Engineering Conference* 169-172. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4685666.

Yoder, Joseph, and Jeffrey Barcalow. n.d. "Architectural patterns for enabling application security." *Urbana* 51:61801. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.3950&rep=rep1&type=pdf.

Zerkle, Dan, and Karl Levitt. 1996. "NetKuang -- A Multi-Host Configuration Vulnerability Checker. "in *in Proceedings of the 6th USENIX Unix Security Symposium*.

Capítulo 9

Desenvolvimento de Gerenciador de Conteúdo com Tecnologia Web 2.0

Cintia Carvalho Oliveira, Daniele Carvalho Oliveira, Cleber Ferreira Oliveira, Renan Gonçalves Cattelan, João Nunes de Souza

Abstract

This paper introduces the concepts and practices related to the development of content management system CMS considering interactive features. We present object-oriented technologies PHP, with MySQL database and Ajax with jQuery. This work consists of theoretical exposition and practical exercises, providing a comprehensive tool for developing Web 2.0 applications.

Resumo

Este trabalho tem como objetivo apresentar os conceitos e práticas relacionadas ao desenvolvimento de gerenciadores de conteúdo (CMS – Content Management System) com recursos interativos. Para tal, são apresentadas as tecnologias PHP orientada a objetos, com banco de dados MySQL e Ajax com jQuery. São apresentados, portanto, os conceitos fundamentais para o desenvolvimento de aplicativos Web 2.0.

9.1. Introdução

O objetivo deste trabalho é apresentar, de forma resumida, tecnologias que apóiam a criação de um CMS (*Content Management System* – Sistema Gerenciador de Conteúdo). Para atender tal objetivo, será criado um CMS, "simples", que consiste de um portal de receitas, no qual cada receita possui uma ferramenta de chat on-line para a participação dos leitores.

9.2. Fundamentos de PHP Orientado a Objetos com o Design Pattern MVC

Para desenvolver um CMS, será utilizada a linguagem PHP (PHP: *Hypertext Preprocessor)*. O PHP é uma linguagem de programação adequada para gerar conteúdo dinâmico na Web. Ela é uma linguagem de scripts, que é embutida em documentos HTML (*HyperText Markup Language*).

Além disso, a linguagem PHP é interpretada por servidores web, como o Apache, com módulo processador de PHP. Dessa forma, a maior parte das tarefas executadas por um programa PHP é invisível ao usuário final. O código da linguagem PHP é interpretado no servidor, gerando uma página HTML, que então é enviada ao usuário.

Finalmente, as razões da escolha da linguagem PHP são, fundamentalmente, as seguintes: Ela é uma linguagem open-source, que tem suporte a praticamente todos os bancos de dados existentes no mercado. Qualquer algoritmo desenvolvido utilizando CGI *(Common Gateway Interface)* pode, também, ser desenvolvidos utilizando PHP. Alguns exemplos são: coleta de dados de formulários, geração de páginas dinamicamente e envio e recebimento de *cookies*. Além disso, PHP é uma linguagem que suporta protocolos como: IMAP, SNMP, NNTP e POP3

Introdução ao PHP orientado a objetos

A orientação a objetos é um paradigma de programação diferente das linguagens estruturadas, onde o conjunto de procedimentos e variáveis nem sempre são agrupados de acordo com o contexto. O paradigma orientado a objetos lida com objetos, que são estruturas representativas de artefatos de nosso dia-a-dia.

Apesar de não ser o foco principal deste trabalho, alguns fundamentos da conceituação de orientação a objetos são apresentados a seguir. Isso, porque tais conceitos serão referenciados ao longo deste trabalho.

Para compreender a orientação a objetos, é fundamental entender o conceito de classes e objetos. Considere, por exemplo, alguns objetos como uma mesa, um computador ou um cachorro. Todos estes objetos compartilham duas características: propriedade e comportamento. O cachorro, por exemplo, tem como propriedades a sua cor, raça, e tamanho. Como comportamento ele come, dorme, late, etc. Objetos, então, são entidades (concretas ou conceituais) do mundo real que tem uma identidade. Ou seja, cada objeto é único, mesmo que outros objetos tenham as mesmas características. Todas as pessoas, por exemplo, possuem características iguais, mas cada uma tem uma identidade. Cada pessoa é um objeto diferente das demais, pois possui nome, CPF, endereço e seu comportamento pode diferir do comportamento das outras. A Figura 1 apresenta exemplos de objetos. O aluno de nome Fernando Soares é um objeto, assim como a professora de nome Regina Campos.

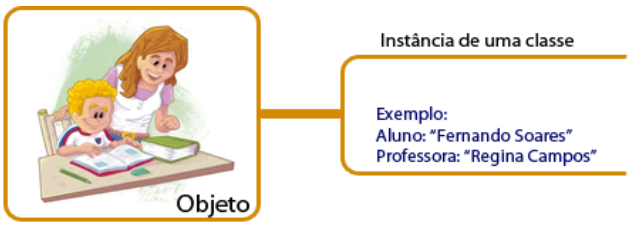


Figura 1 – Exemplo de Objetos

Portanto, existem muitos objetos similares, que possuem características e comportamento semelhantes. Como as pessoas, de modo geral, possuem comportamentos e propriedades semelhantes, é possível definir **Pessoas** como uma classe, e **Fernando Soares** uma instância pertencente à classe **Pessoas.** Nesse contexto,

uma instância é uma ocorrência particular de um objeto de uma classe. A instância possui um estado particular, independente de outras instâncias da mesma classe.

Uma classe é uma estrutura estática que define um tipo de dados. A classe pode conter atributos (variáveis, em analogia à programação estruturada) e funções (métodos) para manipular os atributos. O objeto instanciado contém exatamente a mesma estrutura e as propriedades de sua classe pai. No entanto, a sua estrutura é dinâmica, seus atributos podem mudar de valor durante a execução do programa. Ou seja, a classe é um molde, de onde os objetos são gerados. A Figura 2 apresenta exemplos de classes: a classe **Alunos** e a classe **Professores**.

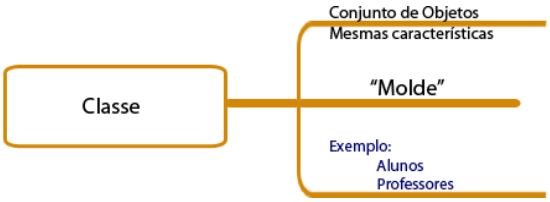


Figura 2 – Exemplos de classes

A Figura 3 mostra a classe **Funcionário,** implementada em PHP.

```php
<?php
class Funcionario {
    var $nome, $salario;

    function Funcionario($nome, $salario){
        $this->nome = $nome;
        $this->salario = $salario;
    }

    function aumentoSalario($valor){
        $this->salario += $valor;
    }
}
?>
```

Figura 3 – Classe Funcionário

E a Tabela 1 descreve os detalhes do código da Figura 3.

Tabela 1 - Código comentado

Linha(s)	Comentário
1 e 14	Abertura e fechamento do código PHP
2	Definição da Classe **Funcionário**. A palavra-chave `class` indica a declaração de classe
3	`var` declara os atributos da classe, a classe **Funcionário** possui dois atributos nome e salário
5	Método construtor da classe **Funcionário** deve possuir o mesmo nome da classe. Este método é executado sempre que uma instância da classe for criada.
6, 7 e 11	Quando a classe referencia seus próprios métodos e atributos, deve-se utilizar a palavra-chave `$this`. O operador `->` faz a referência ao método e atributo.

A Figura 4 apresenta, em PHP, como utilizar a classe criada na Figura 3, instanciando um objeto.

```php
<?php
    include "Funcionario.php";

    $func = new Funcionario("Pedro Cabral", 1200);
    echo("O funcionário ". $func->nome . "recebe R$". $func->salario);
    $func->aumentoSalario(200);
    echo ("Novo salário: " . $func->salario);
?>
```

Figura 4 – Instanciando um objeto da Classe Funcionário

A Tabela 2 comenta o código da Figura 4. O Resultado da execução do código da Figura 4 é:

O funcionário Pedro Cabral recebe R$1200
Novo salário R$1400

Tabela 2: Código comentado

Linha(s)	Comentário
2	A definição da classe deve estar disponível no script ou página PHP que utiliza a classe (comandos `include` ou `require` importam a classe para o script que fará uso dele)
4	Um objeto da classe deve ser instanciado pelo operador `new`.
6	Chamada ao método `aumentoSalario` do objeto `$func`.

Além disso, é possível criar constantes de classe e propriedades estáticas, que armazenam valores comuns a todos os objetos da mesma classe. As constantes são valores fixos, ou seja, não podem ser alterados. E as propriedades estáticas são atributos dinâmicos como as propriedades de um objeto, mas estão relacionados à classe. Esses atributos são acessados externamente pela sintaxe a seguir: `NomeDaClasse::NomeDaConstOuProp`, E dentro da classe, eles são referenciados da seguinte forma: `self::NomeDaConstOuProp`.

Também é possível criar métodos de classe (métodos estáticos), que podem ser chamados diretamente da classe, não sendo necessária a criação de objetos. Esses métodos não devem referenciar propriedades internas pelo operador `$this`. Eles podem apenas referenciar outros métodos estáticos ou propriedades estáticas. Os métodos de classe são referenciados da seguinte forma: `NomeDaClasse::NomeDoMétodo`. A Figura 5 apresenta um exemplo de um método estático. A Tabela 3 contém o comentário do código da Figura 5.

```php
<?php
class Funcionario {
    /* ... */

    static function listarFunc(){
        $bd = new BD();
        $funcs = array();
        $bd->consultar("SELECT * FROM tabela_funcionarios");

        while($linha = mysql_fetch_assoc($bd->resultadoSQL)){
            $funcs['nome_func']['salario_func'];
        }

        return $funcs;
    }
    /* ... */
}
?>
```

Figura 5 – Exemplo de Método Estático

Tabela 3: Código comentado

Linha(s)	Comentário
5	A definição do método estático deve possuir a palavra-chave `static`
6	A classe **BD** é responsável por fazer a conexão com o Banco de Dados e consultas
8	Método consultar da classe **BD** faz a pesquisa do SQL
10	Na variável da classe **BD** `resultadoSQL` está o resultado da consulta da linha 8.

A Figura 6 exemplifica o uso do método estático. A Tabela 4 o comentário do código.

```
1  <?php
2      include_once "funcionario.php";
3
4      $lista = Funcionario:listaFunc();
5
6      foreach($lista as $nome => $salario){
7          echo "O funcionário " . $nome . "recebe R$" . $salario . "<br />";
8      }
9  ?>
```

Figura 6 – Uso do Método Estático

Tabela 4: Código comentado

Linha(s)	Comentário
4	A chamada ao método estático é feita utilizando o operador `::`
6 a 8	O resultado da chamada do método estático é percorrido e impresso na tela

As classes são responsáveis por um assunto específico e possuem "responsabilidades" sobre o seu conteúdo. A classe deve proteger e gerenciar o seu conteúdo através de mecanismos como o encapsulamento, que provê a proteção de acesso ao conteúdo de um objeto. Dessa forma, como é responsabilidade dos métodos da classe tratar seus atributos, qualquer acesso a eles deve ser feito pelos métodos da classe. Assim, as propriedades da classe nunca são acessadas, diretamente, de fora dela. Tal propriedade é, também, chamada de ocultamento de informações. Nesse contexto, o uso do objeto é feito apenas pelo conhecimento da sua estrutura externa. E qualquer mudança da estrutura interna de um objeto não deve afetar o seu acesso externo.

A orientação a objetos e o encapsulamento de classes permitem uma maior organização do sistema. Considerando que cada classe representa uma característica distinta do sistema, então é possível a reutilização de partes do código. E reusar o código possibilita maior agilidade na produção de novos softwares, pois tal fato evita duplicações e reescritas desnecessárias.

Outro recurso da orientação a objetos que, também, possibilita a reutilização de código é a herança. A herança é o compartilhamento de atributos e comportamentos entre as classes de uma mesma hierarquia. A classe **Pessoas**, por exemplo, pode ter atributos em comum com **Professores** e **Alunos**. Cada uma destas classes, que representam profissões, possui características distintas. O professor tem uma formação, ensina, aplica avaliações. E o aluno faz lições, presta atenção às aulas, faz provas. Mas, mesmo com estas diferenças, tais classes possuem semelhanças como: ter nome e endereço, andar, dormir, etc. A classe pai, **Pessoas**, e as classes filhas: **Professores,** e **Alunos**. Todas as características da classe pai serão herdadas pelas classes filhas, ou seja, os elementos da classe **Professores** têm nome, endereços, andam e dormem. Da mesma forma, os elementos de **Alunos**. A Figura 7 apresenta o uso da herança na criação da subclasse **Professor,** que herda as propriedades da classe **Funcionário.**

```php
<?php
include_once "Funcionario.php";

class Professor extends Funcionario{

    var $disciplina;

    function Professor($nome, $salario, $disciplina){
        $this->Funcionario($nome, $salario);
        $this->disciplina = $disciplina;
    }

    function alterarDisciplina($disciplina){
        $this->disciplina =  $disciplina;
    }
}
?>
```

Figura.7 – Subclasse Professor

O código apresentado na Figura 7 é comentado na Tabela 5.

Tabela 5: Código comentado

Linha(s)	Comentário
2	A definição da subclasse deve incluir a definição da superclasse; utilize o comando `include_once` (ou então `require_once`) para evitar problemas
4	`extends` informa que a classe `Professor` é uma subclasse de `Funcionario`
9	O construtor da superclasse deve ser chamado explicitamente pelo construtor da subclasse

Na Figura 8, é apresentado um código que exemplifica a utilização de uma subclasse. O resultado da execução do código da Figura 8 é:

O professor Regina Campos ministra a disciplina Engenharia de software e recebe R$2500
Novo salário R$2800
Nova disciplina: Banco de Dados

```php
<?php
    include_once "Funcionario.php";
    include_once "Professor.php";

    $prof = new Professor("Regina Campos",2500, "Engenharia de Software");
    echo("O professor ". $prof->nome . "ministra a disciplina" . $prof->
disciplina . " e recebe R$". $prof->salario ."<br>");
    $prof->aumentoSalario(300);
    echo ("Novo salário: " . $func->salario . "<br>");
    $prof->alterarDisciplina("Banco de Dados");
    echo ("Nova disciplina: " . $prof->salario);
?>
```

Figura 8 – Uso da Subclasse Professor

Na Tabela 6, o código da Figura 8 é comentado.

Tabela 6: Código comentado

Linha(s)	Comentário
2 e 3	A definição da superclasse e da subclasse devem estar disponíveis no script ou página PHP que utiliza a classe (`include_once` ou `require_once` para evitar problemas)
5	Um novo objeto `Professor` é criado, o método construtor da classe `Professor` é chamado com os parâmetros “Regina Campos”, 2500 e “Engenharia de Software”
6	Observe que o objeto `Professor` criado possui os atributos `nome` e `salário`, que estavam declarados na classe `Funcionário`
7	O objeto `Professor` criado também possui o método `aumentoSalario`

Como descrito nos exemplos anteriores, o uso de classes em PHP facilita a programação nos seguintes aspectos: permite utilizar uma sintaxe mais simples para operações complexas que requerem grande quantidade de argumentos; programas complexos são mais facilmente gerenciáveis e manuteníveis; fomenta a reutilização de código, por meio da criação de componentes genéricos; e permite a substituição de componentes de um sistema, através dos chamados plug-ins.

9.3. Design Pattern MVC (Model, View, Controller)

O MVC (*Model, View, Controller*) é um padrão de projeto que utiliza três camadas, cada uma responsável por uma função. São definidas as três camadas: Modelo, Visão e Controle, que são detalhadas mais adiante.

A Figura 9 apresenta um documento com programação PHP tradicional, onde o código PHP está misturado com marcações HTML. No exemplo, é executada uma consulta ao banco de dados em busca de uma lista dos visitantes cadastrados no portal e, então, o resultado é impresso. Nesse caso, se em outra página fosse necessário imprimir, novamente, uma lista de visitantes, seria necessário reescrever o trecho PHP/SQL. Tal fato torna difícil a manutenção do código e deve ser evitado.

```
1  <h1> Visitantes do Portal </h1>
2  <?php
3      $conexao = mysql_connect('dominio', 'usuario', 'senha');
4      mysql_select_db('banco_dados', $conexao);
5
6      $dados = mysql_query('SELECT nome FROM visitante');
7
8      echo "<ul>";
9      $i = 1;
10     while($usuario = mysql_fetch_array($dados)){
11         echo "<li> Visitante nº" . $i . ": " . $usuario['nome'] . "<//li>";
12         $i++;
13     }
14     echo "</ul>";
15 ?>
```

Figura 9 – Código PHP com HTML

A Tabela 7 apresenta o comentário do código da Figura 9.

Tabela 7: Código comentado da Figura 9.1.9

Linha(s)	Comentário
3 a 6	Conexão com banco de dados e consulta ao nome de todos os visitantes
10 a 12	Impressão dos nomes dos visitantes.

Para facilitar a manutenção e reaproveitamento do código, é necessário dividir o script. Nesse caso, é proposto um documento para a lógica de negócios e o outro para a apresentação. Então, cada arquivo se torna responsável por uma função.

A Figura 10(a) apresenta o arquivo *controle/lista_visitantes.php*, que estabelece a conexão com o banco de dados, consulta todos os nomes dos visitantes e preenche um arranjo (*array*) com os dados. Por outro lado, a Figura 10(b) apresenta o arquivo *visão/lista_visitantes.php*, no qual o arranjo preenchido pelo arquivo anterior é percorrido para a impressão. Nesse caso, esse arquivo se preocupa somente com a apresentação. O arranjo é o resultado da lógica de programação criado pelo arquivo *controle/lista_visitantes.php*.

Essa arquitetura é denominada ***View/Controller*** (*Visão/Controle*), na qual o arquivo *controle/lista_visitantes.php* é o controle e o arquivo *visão/lista_visitante.php* é a visão. A Figura 11 apresenta a arquitetura ***View/Controller*.**

```php
<?php
    $conexao = mysql_connect('dominio', 'usuario', 'senha');
    mysql_select_db('banco_dados', $conexao);

    $dados = mysql_query('SELECT nome FROM visitante');
    $visitante = array();

    while($usuario = mysql_fetch_array($dados)){
        $visitante[] = $usuario['nome'];
    }
?>
```

(a)

```php
<?php include_once "../controle/lista_visitantes.php"; ?>
<h1> Visitantes do Portal </h1>
<ul>
<?php
    foreach ($visitante as $i => $nome){
        echo "<li> Membro nº $i: $nome </li>";
    }
?>
</ul>
```

(b)

Figura 10 – Modelo *View/Control*

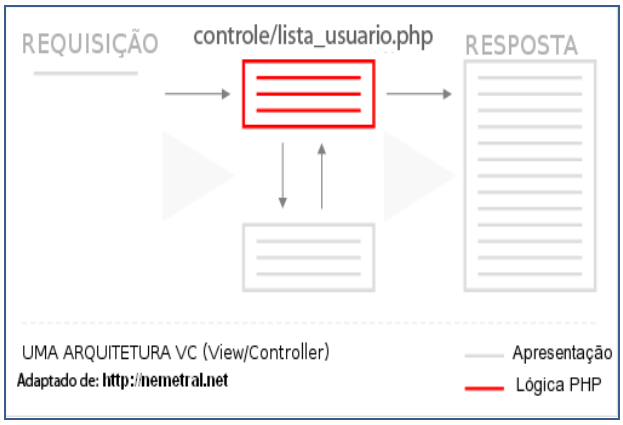


Figura 11 – Arquitetura *View/Controller*

O padrão de projeto **MVC** define a divisão das aplicações em 3 camadas, **Modelo**, **Visão** e **Controle**.

Modelo: O **Modelo** é a camada de acesso aos dados. Ela manipula os dados junto ao banco de dados, no qual consulta, adiciona, altera ou remove dados quando solicitado. A função `listaVisitantes()`, `por exemplo`, retorna um arranjo - como na Figura 10(a), nas linhas 8 a 10 - com todos os visitantes de um site, ou `cadastrarVisitante($nome, $email)` para adicionar um novo usuário no banco de dados.

Visão: A **Visão** é a camada responsável pela apresentação. É nessa camada que é gerada a saída. No exemplo, tal camada processa o arranjo que foi obtido na saída da função `listaVisitantes()`. Percorre o arranjo, imprimindo na tela o HTML final da página. Como resultado, são exibidos os nomes de todos os usuários cadastrados no portal, como indicado nas linhas 5 a 7 da Figura 10(b).

Controle: A camada de **Controle** se destina a coordenar todo o processo. Ela analisa as requisições e executa o modelo correspondente para obter os dados. No exemplo, ela chama a função `listaVisitantes()`, a qual retorna o arranjo com a lista dos visitantes. A camada de **Controle** passa o arranjo obtido para a camada de **Visão**, que o percorrerá imprimindo os nomes. Por fim o **Controle** exibe a saída.

A Figura 12 apresenta o esquema de funcionamento do MVC, no qual é explicitada a interação entre as camadas. Tal interação ocorre desde o momento que houve a requisição HTTP, por meio do navegador, até o retorno com a saída HTML.

Nesse sentido, a arquitetura MVC provê mais flexibilidade na construção da aplicação. Além disso, na arquitetura MVC, o acesso aos dados é de responsabilidade da camada Modelo. Dessa forma, a implementação de uma função, como, por exemplo, `listaVisitantes()`, que pertence à camada **Modelo**, executa uma consulta em banco de dados ou arquivos XML. Nesse contexto, tal função é solicitada pela camada **Controle**, que faz a chamada da referida função e, simplesmente, espera pelo resultado de sua execução. Nesse caso, a saída é um arranjo com os nomes dos visitantes.

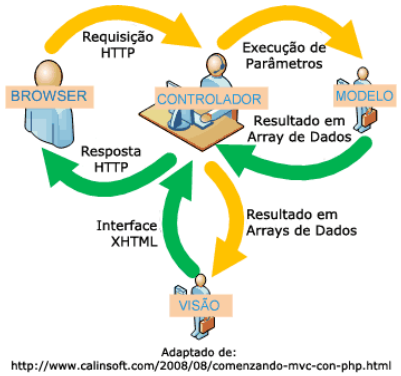


Figura 12 – Funcionamento MVC

A apresentação é responsabilidade da camada **Visão**. A apresentação pode ser, por exemplo, um *feed* RSS, ou uma página HTML.

Além disso, uma função na camada **Controle** pode escolher qualquer tipo de apresentação que a camada **Visão** oferece. Tal característica é uma vantagem da arquitetura MVC, pois apresenta aspectos de reusabilidade. Isso ocorre porque é possível reutilizar o mesmo modelo, com a mesma lógica e criar duas saídas diferentes (ambas imprimem a lista de visitantes, por exemplo).

Na arquitetura MVC, a camada **Controle** pode ser chamada diretamente. Nesse caso, o navegador referencia a função desejada, que pertence à camada **Controle.** Outra opção é aquela em que o navegador acessa um único ponto de entrada do sistema, que, geralmente, é denominado roteador. Nesse caso, o navegador informa ao roteador a função desejada pelo usuário. Então, o roteador escolhe a função, da camada **Controle,** adequada para atender tal solicitação.

A Figura 13 apresenta um esquema de funcionamento da arquitetura MVC com a utilização de um roteador.

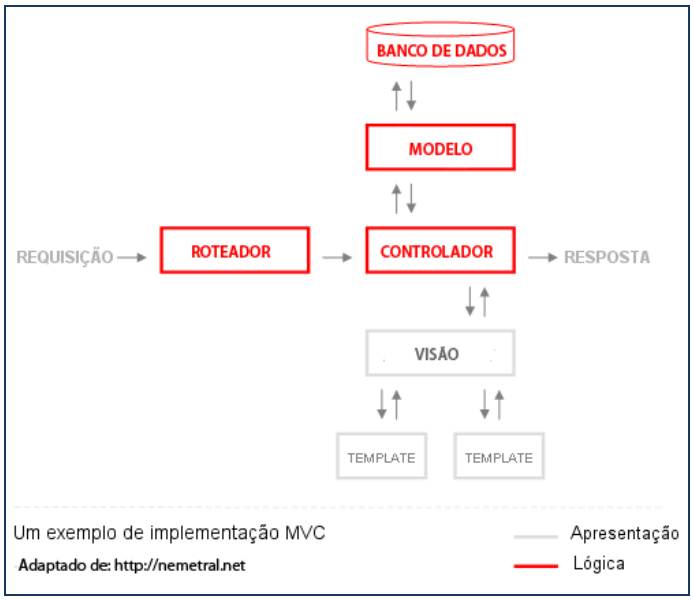


Figura 13 – Arquitetura MVC com o roteador

Programação da interface em camadas

Como a camada **Visão** contém as características: estilo, estrutura e comportamento, então a interface do sistema pode ser programada, também, em camadas.

Nesse contexto, o estilo é programado em CSS, a estrutura em HTML e o comportamento em JavaScript.

O HTML permite a inclusão de marcações e atributos para configuração de estilos. Um exemplo é adicionar alinhamento em uma tag de cabeçalho `<h1>`.

A Figura 14 apresenta um documento no qual o conteúdo está sendo formatado com estilos incluídos no próprio documento XHTML. A figura 15 é o resultado desse documento.

```
1  <html>
2  <head>
3      <title>Documento com Marcação de Apresentação</title>
4  </head>
5  <body>
6      <h1 align="right">Marcação de Apresentação</h1>
7      <p align="right"><font face="Arial, Helvetica, sans-serif" size="2" color="blue">Aqui o
   conteúdo será alinhado à direita, num tipo de letra sans-serif de tamanho 2, e a azul.</font></p>
8  </body>
9  </html>
```

Figura 14 - Documento XHTML com marcações de apresentação

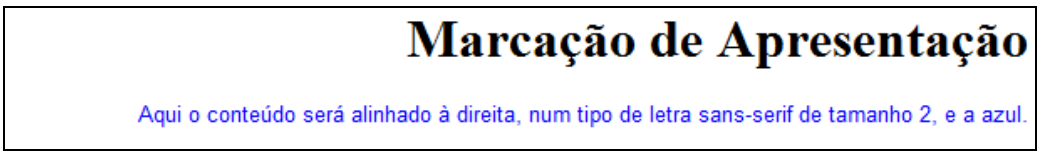


Figura 15 - Interface do Documento com marcações de apresentação

A Tabela 8 comenta o código da Figura 14.

Tabela 8 - Código comentado

Linha(s)	Comentário
1 e 9	Abertura e fechamento do documento HTML
2 a 4	Cabeçalho do documento com o título da página
5 e 8	Abertura e fechamento do corpo do documento HTML
6	O elemento `<h1>` define um cabeçalho com tipo de letra grande, e o atributo `align` define que o alinhamento do texto fica à direita
7	A marcação `<p>` define um parágrafo, e `align` um alinhamento à direita. E `<font>` é uma marcação em desuso, ela configura o estilo da fonte do texto demarcado por ela, o atributo face define o tipo de letra, `size` o tamanho e `color` a cor da fonte.

No caso do documento anterior, quanto maior for o documento, as estruturas e o estilo (toda marcação ou atributo que define o desenho, o estilo e não a estrutura) maior, mais complexo e confuso é o código. Nesse caso, é necessário um código fonte estruturado, que separa o conteúdo e sua estrutura, dos estilos e do comportamento.

A figura 16 apresenta um documento HTML, que contém somente a estrutura, sem nenhum estilo e com configurações padrões. Na Figura 17 é apresentada a interface definida pelo código da Figura 16.

```
1  <html>
2  <head>
3      <title>Documento com Estrutura</title>
4  </head>
5  <body>
6      <h1>Documento Estruturado</h1>
7      <p>A não ser que lhe seja aplicada uma folha de estilo, este parágrafo será apresentado com os
    valores default.</p>
8  </body>
9  </html>
```

Figura 16 - Documento com estrutura

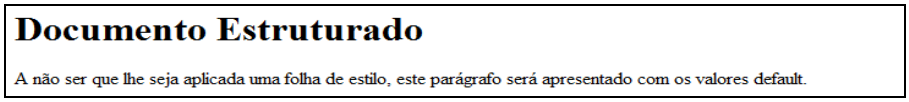


Figura 17 - Interface do documento

Na Figura 18 a estrutura e o estilo são separados. O resultado no navegador é mostrado na figura 19.

```
1  <html>
2  <head>
3      <title>Documento com estrutura e estilo no mesmo documento</title>
4  <style type="text/css">
5  <!--
6  h1 {
7      text-align: right;
8  }
9
10 p {
11     text-align: right;
12     font-family: Arial, Helvetica, sans-serif;
13     font-size: 16px;
14     color: blue;
15 }
16 -->
17 </style>
18 </head>
19 <body>
20     <h1>Documento com estrutura e estilo juntos</h1>
21     <p>Neste documento temos a estrutura da página e o estilo definidos no mesmo documento, porém
    separados.</p>
22 </body>
23 </html>
```

Figura 18 - Documento com estrutura e estilo no mesmo documento

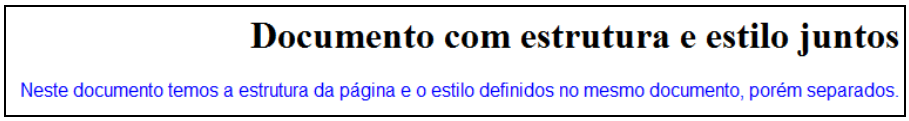


Figura 19 - Resultado no navegador

Na Tabela 9, o código da Figura 18 é comentado.

Tabela9 - Código comentado

Linha(s)	Comentário
4 e 17	Abertura e fechamento das definições de estilo em CSS
6 a 8	Customização do estilo das marcações `h1`, definindo alinhamento à direita.
10 a 15	Customização do estilo das marcações `p`, definindo um alinhamento à direita, tipo de fonte Arial, tamanho 16 e cor azul.

No documento anterior a estrutura e o estilo ainda não estão separados. Então, o próximo passo é separar estilo e estrutura em dois documentos diferentes. Procedendo dessa forma, são obtidos arquivos separados e especializados. Ou seja, a folha de estilo, Figura 20(a) se chama *apresentacao.css*. Nela só existem as definições de elementos e de como ficará sua apresentação. Por outro lado, no arquivo *apresentacao.html* da Figura 201(b) há somente a estrutura.

```
1  h1 {
2  text-align: right;
3  }
4
5  p {
6  text-align: right;
7  font-family: Arial, Helvetica, sans-serif;
8  font-size: 16px;
9  color: blue;
10 }
```

(a)

```
1  <html>
2  <head>
3      <title>Documento com estrutura e estilo</title>
4      <link type="text/css" rel="stylesheet" href="apresentacao.css">
5  </head>
6  <body>
7      <h1>Documento com estrutura e estilo em dois documentos</h1>
8      <p>Neste documento temos a estrutura da página em um arquivo e o
   estilo definidos em outro. </p>
9
10 </body>
11 </html>
```

(b)

Figura 20 – Estilo e estrutura em arquivos separados

Na Tabela 10 o código da Figura 20(b) é comentado.

Tabela 10: Código comentado

Linha(s)	Comentário
4	Aplicação da folha de estilo *apresentacao.css*.

Uma estrutura HTML pode ser apresentada de diversas maneiras, dependendo somente da folha de estilo aplicada. Na Figura 20(b), a linha 4 é um ligação com o arquivo *apresentacao.css*. Entretanto, no lugar desse arquivo pode haver outro que tem outros tipos de estilos para os elementos do documento *apresentacao.html*.

Como o comportamento é programado em JavaScript, neste trabalho é considerada a biblioteca jQuery para efetuar tal programação. A biblioteca jQuery customiza comportamentos para elementos HTML. Assim, como na programação do estilo, a programação do comportamento também pode ser separada da estrutura. Logo, para a programação da interface em camadas, é necessário separar em arquivos distintos a **Estrutura**, o **Estilo** e o **Comportamento**. A seguir, é descrito o que deve conter cada uma dessas camadas.

- **Estrutura**: Essa camada contém: marcações padrão de HTML, tais como `<html>`, `<head>`, `<title>`, `<body>`; marcações que representam estruturas, tais como parágrafos, quebras de linha, listas, divisões, tabelas e formulários.
- **Estilo**: Essa camada contém: tudo que diz respeito ao desenho e não à estrutura, tais como alinhamentos, cor, fontes, bordas etc.
- **Comportamento**: Essa camada contém: programas em JavaScript. Neste trabalho, é utilizada a biblioteca jQuery.

Dada a arquitetura MVC e a programação da interface em camadas, tais conceitos são utilizados a seguir para criar um Sistema Gerenciador de Conteúdo.

9.3. CMS (Content Management System)

Um sistema gerenciador de conteúdo (*CMS - Content Management System*) define uma forma de criar e manter páginas web. O CMS é constituído de um conjunto de ferramentas que conferem agilidade, segurança e confiabilidade requeridas para o tratamento da informação.

Ferramentas do CMS permitem o gerenciamento de conteúdo proveniente de diversas fontes. Elas podem ser destinadas a diversos dispositivos como *browsers* ou celulares. Além disso, elas possuem como vantagens a redução do custo com a atualização de conteúdo, aumento de canais de publicação de conteúdo, padronização da informação e aumento da eficiência das equipes de TI.

A idéia do CMS é separar o gerenciamento do conteúdo do *design* gráfico das páginas que o apresentam. Para isso, o design é visto como o molde, um esqueleto, enquanto o conteúdo é armazenado em banco de dados ou arquivos separados. Como o *design* é visto como um molde, o administrador, ou autor de um site, não precisa se preocupar com detalhes a respeito da interface. É necessário estar atento somente ao conteúdo que é, nele, exibido.

A ferramenta CMS permite que o autor insira conteúdo através de algum meio como: formulários, *upload* de arquivo etc. E tal informação deve ser armazenada em um repositório de dados. Assim, quando um usuário visita o site, o sistema requisita a informação na base de dados, formata, insere na interface e apresenta o resultado ao usuário. Além disso, o CMS permite a inserção, alteração, exclusão e busca de conteúdo em tempo real, sem necessidade de conhecimento de linguagem de programação (PHP, Java etc) de marcação (HTML), e nem de intermediadores para a inserção deste conteúdo.

Alguns tipos de CMS são, livremente, disponibilizados. Como exemplos, há o Drupal, Joomla e Wordpress. De acordo a literatura, o Wordpress é utilizado por pessoas que necessitam de ferramentas para publicar conteúdo e blog. Por outro lado, o Drupal permite extensa customização/programação. E o Joomla possui maior facilidade de personalização, além de possuir quantidade e qualidade de componentes e extensões.

O CMS possui duas sessões, o **Frontend** e o **Backend,** que são descritas na Figura 21.

1. **Frontend** – O **Frontend** atende os usuários do site
2. **Backend** – O **Backend** atende os administradores e editores.

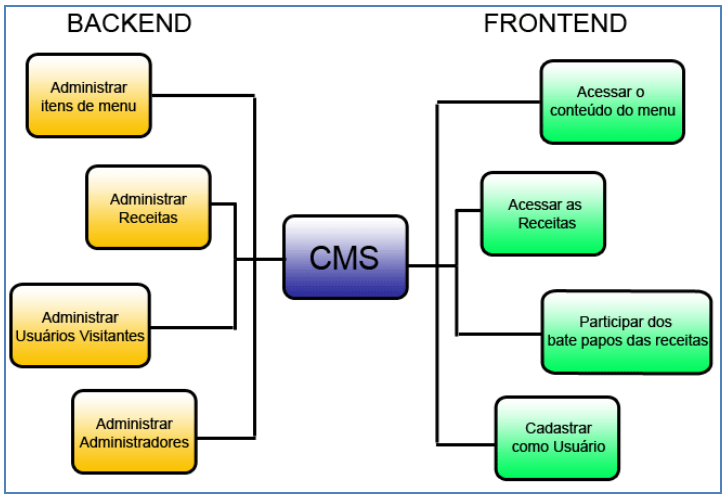


Figura 21 – CMS de Portal de Receitas

Conforme a Figura 21, o **Backend** oferece ferramentas para o administrador, como é detalhado a seguir:

- Administrar itens de menu: Essa funcionalidade permite adicionar/alterar/remover itens de menu. Eles podem ser públicos ou privados;
- Administrar Receitas: Essa funcionalidade permite adicionar/alterar/remover receitas que podem ser públicas ou privadas;
- Administrar Usuários Visitantes: Essa funcionalidade permite verificar novas inscrições de visitantes, podendo excluir visitantes;
- Administrar Administradores: Essa funcionalidade permite adicionar/remover novos administradores do portal.

Conforme a Figura 21, o **Frontend** oferece os recursos que ficarão acessíveis para o usuário, como é detalhado a seguir:

- Acessar o conteúdo do menu: Essa funcionalidade permite acesso ao conteúdo do menu. Se ele for privado, o usuário poderá acessar se ele estiver autenticando no site,
- Acessar as receitas: Essa funcionalidade permite acessar as receitas públicas. Se o usuário estiver autenticado no site, ele poderá acessar as receitas privadas.
- Participar dos bate papos das receitas: Essa funcionalidade permite ao usuário participar de um bate papo
- Cadastrar como Usuário: Essa funcionalidade permite acessar conteúdo privado.

Usabilidade em CMS

A usabilidade de alguma coisa é a característica que permite a sua utilização eficiente, efetiva e satisfatória. No CMS, o objetivo é criar um sistema que seja fácil de entender e simples de usar, mesmo por usuários leigos. Além disso, deve ser uma ferramenta intuitiva e que guie o usuário no seu uso.

Inicialmente, no projeto de um CMS, para que ele seja usável, é necessário definir o objetivo do sistema. Nesse casso ele pode ser utilizado atendendo uma, ou mais, das características as seguir.

- Escrita e estruturação de conteúdo de forma consistente;
- Customizar conteúdo para diferentes usuários;
- Customizar conteúdo para diferentes mídias;
- Distribuir conteúdo dinamicamente;
- Armazenar conteúdo e acessá-lo;
- Recuperação e reuso de conteúdo

Após a definição do objetivo do sistema, é necessário, ainda, responder algumas questões:

- Como o conteúdo será acessado?
- Como o conteúdo deverá ser identificado para que os usuários encontrem o que procuram?
- Como o conteúdo deve ser estruturado para corresponder aos objetivos do CMS?

Por fim para que um CMS seja usável é necessário avaliar as necessidades do sistema, seus usuários e o seu conteúdo, como definido a seguir:

- Avaliação das necessidades: As necessidades de um CMS são obtidas respondendo a questões do tipo: Por que um CMS é necessário? O que gerenciar conteúdo significa na presente situação e o que se deseja que ele faça por você? Quais problemas devem ser resolvidos por ele? Quais oportunidades deseja aproveitar?
- Avaliação dos usuários: Os usuários de um sistema devem ser identificados. É útil criar uma matriz usuário/tarefa para definir os tipos de usuários que utilizarão o sistema. A partir dessa matriz é possível traçar estratégias que definirão como os usuários executarão suas respectivas tarefas.
- Avaliação do conteúdo: A avaliação do conteúdo consiste em estruturá-lo de forma a atender às necessidades do usuário. Por exemplo, em uma loja virtual ao acessar um produto quais informações o usuário busca? Em qual ordem elas devem aparecer?

Esses passos permitem ao desenvolvedor do CMS compreender os objetivos e características que o sistema deve possuir, para assim guiar a criação de um sistema usável.

9.4. Web 2.0

A Web 2.0 é um conceito que possibilita expressar o comportamento dos usuários. Essa Web não se trata somente de novas tecnologias, mas do modo como as pessoas se relacionam e como a tecnologia suporta esse comportamento. Para suportar esta interação e colaboração, são necessárias novas técnicas e métodos computacionais. Assim os novos sistemas web devem ser voltados para que os usuários, hoje ativos e participantes, tenham meios de colaborar com o site.

9.5. Notas Bibliográficas sobre as seções 9.1, 9.2, 9.3 e 9.4

No site *http://php.net* existe uma completa documentação da linguagem de programação PHP. Uma abordagem sobre orientação a objetos em PHP pode ser encontrada no livro ***PHP: Programando com Orientação a Objetos*** de Pablo Dall'Oglio.

A base para a criação da seção sobre o padrão de projeto MVC foram três artigos sobre o tema publicados no site *http://nemetral.net/* do autor Nemetral. O site *http://www.calinsoft.com* também trás uma abordagem para o ensino do padrão, com referência a alguns frameworks que o implementam tais como o CakePHP e o CodeIgniter, entre outros.

Os autores Júlio Pereira e Marcello Bax no artigo intitulado ***Introdução à Gestão de Conteúdos*** abordam sobre o uso de Gerenciadores de Conteúdo e as suas vantagens.

Em *http://www.portal2web.com.br/software-livre/drupal-x-joomla-x-wordpress-qual-o-melhor-cms.html* temos uma comparação entre alguns dos CMS mais utilizados atualmente.

O Dominic Kristaly em seu artigo intitulado ***Web 2.0 technologies in web application development*** aborda a respeito do conceito de Web 2.0, sua aplicação em CMS, e descreve um sistema gerenciador de conteúdo desenvolvido chamada Butterfly. Pamela Kostur em seu trabalho ***Incorporating Usability into Content Management*** **explica** o que é usabilidade e como ela se aplica no planejamento e desenvolvimento de um sistema gerenciador de conteúdo.

Sobre Web 2.0 o presente trabalho se baseou nas obras de Tim Berners-Lee, Tim O'Reilly e Paul Anderson, ***Weaving the Web: the past, present and future of the World Wide Web by its inventor, What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*** *e* ***What is Web 2.0? Ideas, Technologies and implications for education***, respectivamente.

9.6. jQuery

Introdução

jQuery é uma biblioteca JavaScript que simplifica a manipulação de documentos HTML, o uso de animações e requisições Ajax, propiciando um rápido desenvolvimento de aplicações web. Seu criador foi o desenvolvedor americano John Resig.

O jQuery tem como objetivos facilitar o uso do JavaScript para realizar as seguintes tarefas:

- Adicionar efeitos visuais e animações;
- Acessar e manipular o documento HTML;
- Alterar o conteúdo de uma página sem recarregá-la;
- Prover interatividade;
- Modificar apresentação e estilização de elementos HTML;
- Simplificar tarefas específicas do JavaScript.

A biblioteca jQuery atende os Padrões Web, o que significa que ela é compatível com qualquer sistema operacional e navegador. Essa biblioteca possui as seguintes características:

- Utiliza seletores CSS para localizar elementos em um documento HTML;

- Possibilita a instalação de plug-ins e extensões;
- É indiferente às inconsistências de renderização de navegadores;
- Localiza elementos dentro de um documento de forma precisa;
- Admite programação encadeada;
- É extensível, admitindo a inserção de novas funcionalidades na biblioteca.

Além disso, a biblioteca jQuery é um arquivo JavaScript, que deverá ser linkado à página web que utilizar programação jQuery. A Figura 22 apresenta a linha que faz o link com a biblioteca. Tal link dever ser colocado no cabeçalho do documento, ou seja, entre as marcações `<head></head>`. Já o atributo `src`, da marcação `<Script>`, indica o caminho no qual o arquivo JavaScript está gravado.

```
...
<head>
...
    <script type='text/javascript' src='jquery.js'></script>
...
</head>
...
```

Figura 22 – Link com a biblioteca jQuery

Na Figura 23 é apresentado um código com JavaScript in-line (script dentro de marcação HTML). Tal escrita de código contraria um dos princípios fundamentais dos padrões Web, que trata da separação total entre estrutura, estilo e comportamento.

```
1  ...
2      <style type="text/css" media="all">
3          h1 { color:#090; } /* verde */
4      </style>
5  </head>
6  <body>
7      <h1 id="cor"> Cabeçalho muda de cor </h1>
8      <button type="button" onclick="document.getElementById('cor').style.color = '#F00';">
9          Vermelho
10     </button>
11 ...
```

Figura 23 – JavaScript in-line

A Figura 24 apresenta a interface que corresponde ao código da Figura 23. Quando o botão de nome “Vermelha” é clicado, o método `onclick` do elemento `button` (linha 8) é acionado. Então, o código JavaScript é executado, alterando a cor do elemento cujo `id` é cor, ou seja, a marcação `<h1>` (linha 7).

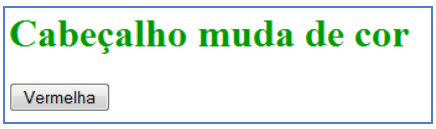


Figura 24 - Interface do arquivo da Figura 23

Essa mesma funcionalidade é representada na Figura 25, com uma marcação HTML isenta de código JavaScript. Isso é desejável, porém seria melhor ainda se fossem separados o estilo, comportamento e conteúdo em documentos diferentes.

A tabela 11 comenta o código da Figura 25.

```
1  ...
2      <style type="text/css" media="all">
3          h1 { color:#090; } /* verde */
4      </style>
5      <script type="text/javascript">
6          window.onload = function(){
7              document.getElementById('btn-vermelho').onclick = mudaCor()
8          };
9          function mudaCor(){
10             document.getElementById('cor').style.color = '#F00'; /* vermelho */
11         };
12     </script>
13 </head>
14 <body>
15     <h1 id="cor"> Cabeçalho muda de cor </h1>
16     <button type="button" id="btn-vermelho">Vermelho</button>
17 ...
```

Figura 25 – Código JavaScript separado do HTML

Tabela 11 – Comentários a respeito do código da Figura 25

Linha(s)	Comentário
6 a 8	Assim que o arquivo terminar de ser carregado (linha 6) será atribuído um evento `onclick` ao elemento que possui como `id` `btn-vermelho`, quando este for clicado a função `mudaCor` é chamada.
9 a 11	Função `mudaCor` quando acionada alterará a cor do elemento cujo `id` for `cor`.

A Figura 26 apresenta o script programado, utilizando a biblioteca jQuery. Como é indicado, o código gerado tem a mesma função do código da Figura 25. Porém, ele é mais simples e, conseqüentemente, mais rápido de programar.

```
1  ...
2      <style type="text/css" media="all">
3          h1 { color:#090; } /* verde */
4      </style>
5      <script type="text/javascript" src="jquery.js"></script>
6      <script type="text/javascript">
7          $(document).ready(function(){
8              $("#btn-vermelho").click(function(){
9                  $("#cor").css("color","#FF0000");
10             });
11         });
12     </script>
13 </head>
14 <body>
15     <h1 id="cor"> Cabeçalho muda de cor </h1>
16     <button type="button" id="btn-vermelho">Vermelho</button>
17 ...
```

Figura 26 – Código jQuery

A tabela 12 comenta o código da Figura 26 e explica algumas das principais estruturas da biblioteca.

Tabela 12 – Código da Figura 26 comentado

Linha(s)	Comentário
5	Link com o arquivo da biblioteca jQuery para poder utilizar a sintaxe jQuery.
6	O `$()` é a função construtora da biblioteca, ela é responsável por encontrar os elementos em um documento. Na linha 6 ela está referenciando o documento como um todo. `ready()` é um método similar ao `window.onload`. Assim `$(document).ready` irá executar quando a estrutura (árvore) do documento terminar de carregar.
8	A função `$()` encontra o elemento cujo `id` (referenciamos o `id` por #) é `btn-vermelho` e atribui um evento `click` a ele.
9	Quando o botão for clicado (linha 8) a função `$()` encontra o elemento cujo `id` for `cor` e modifica um atributo CSS, no caso a cor, colocando o código referente à cor vermelha.

Conforme os exemplos anteriores, uma das estruturas mais básicas do jQuery é a função construtora `$()`. Ela é responsável pela localização dos elementos dentro da árvore do documento.

Assim, caso seja necessário, por exemplo, alterar a cor de fundo de todos os elementos `<div>` para a cor vermelha, então a sintaxe a ser utilizada deve ser aquela apresentada na Figura 27.

```
$("div").css("background","red");
```

Figura 27 – Todos os elementos com fundo vermelho

Mas, encontrar todos os elementos de um tipo pode não ser aquilo desejado pelo programador. Nesse caso, pode ser que o programador deseja localizar apenas uma ocorrência específica de um elemento. Se há um elemento, como na Figura 28, cujo `id` é `xyz`, o jQuery o localiza, como mostrado na Figura 29.

```
<p id="xyz"> Texto do parágrafo </p>
```

Figura 28 – elemento <p> identificado por xyz

```
$('#xyz');
```

Figura 29 – Função construtora localiza elemento com id igual à xyz

Encadeamento de funções

O encadeamento de funções da biblioteca jQuery é o ordenamento em uma única linha de várias funções. Um exemplo desse encadeamento é mostrado na Figura 30.

```
$('div').children('p').fadeOut().addClass('xyz');
```

Figura 30 – Exemplo de encadeamento

A sintaxe da Figura 30 diz o seguinte:

"Construtor, encontre todos os elementos `parágrafos` que sejam filhos dos elementos `div`, neles aplique um efeito de esmaecimento (`fadeOut`) e adicione a classe CSS `xyz`."

Em jQuery, o encadeamento é possível porque cada método retorna um objeto, que por sua vez pode ser processado pelo método seguinte.

Funções-padrão e seletores jQuery

A biblioteca jQuery possui funções-padrão que podem ser utilizadas em programação, com vários objetivos. Algumas delas são:

» **`.css`**

A função .css adiciona, ou altera, um estilo do elemento associado. Na Figura 31, o primeiro parâmetro indica a propriedade CSS do elemento que se deseja alterar. O

segundo parâmetro é a quantificação ou característica da propriedade. Na Figura 31, seu código altera o tamanho da caixa do bate papo para 200 pixels.

```
$('#chat').css('height','200px');
```

Figura 31 – Função .css

» **$( )**

A função $() é a principal função jQuery. Ela aceita dois parâmetros, como mostrado na Figura 32.

```
$('p',$('#chat')).css('color','green');
```

Figura 32 – Construtor $()

Na Figura 31 o construtor `$()` possui somente um parâmetro (`#chat`) que indica a busca do elemento cujo `id` é `chat`. Na Figura 32 há um segundo parâmetro que é denominado contexto. Ele é utilizado para localizar (no caso do exemplo) elementos `<p>` que estejam dentro de elementos, cujo `id` é `chat`. Se o segundo argumento é omitido, então todos os parágrafos do documento são estilizados na cor verde. O resultado da aplicação da função da Figura 32 na Figura 33 faz com que todos os parágrafos dentro de chat fiquem verdes.

```
<div id="chat">
    <p> Primeiro Parágrafo. </p>
    <p> Segundo Parágrafo. </p>
</div>
```

Figura 33 – Exemplo de documento HTML

As funções a seguir selecionam elementos dentro do documento. Existem várias formas de efetuar esta seleção de acordo com o objetivo do script.

» **$( )**

A função construtora `$()` pode ser empregada como uma função seletora. Quando utilizada na forma da Figura 34 seleciona um elemento cujo atributo `id` é igual a `receita` (elemento presente na linha 3 da Figura 35) e altera seu estilo.

```
$('#receita').css('background','red');
```

Figura 34 - Selecionando elemento com id igual à receita

```
1  <h1> Título da Receita </h1>
2
3  <div id="receita">
4      <p> Corpo da Receita. </p>
5      <p> Autor. </p>
6  </div>
7
8  <form class="form_verde" id="contato">
9      Nome: <input type="text" id="nome" /> <br />
10     E-mail: <input type="text" id="email" /> <br />
11     Texto: <input type="text" id="email" /> <br />
12     <input type="submit" value="Enviar" />
13 </form>
```

Figura 35 - Exemplo de um documento HTML com um formulário

Na Figura 36, o parâmetro da função construtora seleciona um elemento cujo atributo `class` é igual à `form_verde` (elemento presente na linha 8 da Figura 9.2.15) e altera seu estilo.

```
$('.form_verde').css('background','#60F');
```

Figura 36 – Selecionando elemento com class igual à form_verde

O parâmetro da função construtora escrito como no código da Figura 37 seleciona todos os elementos `input` do documento (elementos presentes nas linhas 9 a 12 da Figura 35) e altera seu estilo.

```
$('input').css('background','#FFC');
```

Figura 37 – Seleciona todos os elementos input

A função `$()` escrita como no código da Figura 38 permite definir um conjunto de seletores como argumento.

```
$('#receita, input, .form_verde').css('background','red');
```

Figura 38 – Vários elementos sendo selecionados

» **`:first`**

O seletor `:first` cuja sintaxe é apresentada na Figura 39, seleciona a primeira ocorrência do elemento associado a ele, no caso a linha 2 da Figura 40.

```
$('li:first').css('background','#60F');
```

Figura 39 – Seletor filtro :first

```
1  <ul>
2      <li> Primeiro item </li>
3      <li> Segundo item </li>
4      <li> Terceito item </li>
5      <li> Quarto item </li>
6      <li> Quinto item </li>
7  </ul>
```

Figura 40 – Exemplo de documento HTML com uma lista não ordenada

» **`:last`**

O seletor `:last` seleciona a última ocorrência do elemento associado a ele. A sintaxe do código da Figura 41 altera o estilo do elemento da linha 6 da Figura 40.

```
$('li:last').css('background','green');
```

Figura 41 – Seletor filtro :last

» **`:even`**

O seletor `:even` seleciona as ocorrências de ordem par do elemento associado a ele O código da Figura 42 altera o estilo dos elementos das linhas 2, 4 e 6 da Figura 40, que correspondem aos índices 0, 2 e 4 (a contagem inicia no 0).

```
$('li:even').css('background', 'red');
```

Figura 42 – Seletor filtro :even

» **`:odd`**

O seletor `:odd` seleciona as ocorrências de ordem ímpar do elemento associado a ele. O código da Figura 43 altera o estilo dos elementos das linhas 3 e 5, que correspondem aos índices 1 e 3 (a contagem inicia no 0) da Figura 40.

```
$('li:odd').css('background', 'green');
```

Figura 43 – Seletor filtro :odd

» **`:eq`**

O seletor `:eq` seleciona a última ocorrência do elemento associado a ele. O código da Figura 44 altera o estilo do quarto elemento representado na linha 5 da Figura 40.

```
$('li:eq(3)').css('background', 'yellow');
```

Figura 44 – Seletor filtro :eq

Manipulação de elementos da árvore do documento HTML

A seguir são apresentadas funções jQuery que definem atributos e valores a elementos de um documento HTML.

» **`.attr`**

Se um documento HTML possui uma imagem como no código da Figura 45, a sintaxe do método `.attr` mostrado na Figura 46 captura o valor do atributo `src` da imagem e a armazena na variável `arquivo`.

```
<img src="foto.jpg">
```

Figura 45 – HTML com uma imagem

```
var arquivo = $('img').attr('src');
```

Figura 46 – Método .attr

O método `attr` é utilizado, também, para adicionar atributos e valores a um elemento. Na Figura 47 a sintaxe do método `attr` adiciona ao elemento `img` vários atributos com seus respectivos valores. Na Figura 48 a sintaxe do método `attr` adiciona ao elemento `<img>` somente um atributo e seu valor.

```
$('img').attr({
    border:"10px",
    src: "foto2.jpg"
});
```

Figura 47 – Método .attr adicionando múltiplos atributos

```
$('img').attr('border','10px');
```

Figura 48 – Método .attr adicionando somente um atributo

» **`.removeAttr`**

A função `.removeAttr` acessa o elemento associado a ele e remove o atributo definido em seu parâmetro. No código da Figura 49, o método `removeAttr` remove o atributo `src` do elemento `img`.

```
$('img').removeAttr('src');
```

Figura 49 – Método .removeAttr

» **`.addClass`**

O método `.addClass` atribui ao elemento associado a ele uma classe CSS já definida. Na Figura 50, se a classe CSS `form_verde` existe, então o método `.addClass` atribuirá ao formulário do documento esse estilo.

```
$('form').addClass('form_verde');
```

Figura 50 – Método .addClass

» **`.html`**

A função `.html` captura ou atribui código HTML a um elemento. Na Figura 51 o método `.html` captura o conteúdo do elemento `p` e o atribui à variável `conteudo`. Na Figura 52 a função `.html` atribui o conteúdo da variável `dado` (a variável contém código HTML) a um elemento `div` (exemplo retirado do bate papo do CMS).

```
var conteudo = $('p').html();
```

Figura 51 – Capturando conteúdo html de um elemento

```
$('#chat').html(dado);
```

Figura 52 – Atribuindo conteúdo html a um elemento

Na biblioteca jQuery existem ainda os métodos `.text` e `.val` que são semelhantes ao método `.html`.

Eventos

Eventos são ações realizadas pelos usuários no navegador. Quando o usuário abre uma página, clica em um botão, seleciona uma opção em um campo ou preenche um formulário ocorre um evento. A seguir são apresentados alguns eventos jQuery:

» **`.blur`**

O evento `.blur` é chamado quando um elemento perde o foco. Por exemplo, um usuário ao preencher um formulário segue de campo em campo preenchendo-o com informações, quando ele termina de preencher um campo e passa para outro o campo anterior perde o foco, cedendo-o ao próximo elemento. Assim o evento `.blur` é muito usado em formulários para validar a entrada de um dado em um campo. Na Figura 53 quando o campo perde o foco uma caixa de alerta é acionada.

```
$('input').blur(function() {
    alert("Você acionou o evento blur");
});
```

Figura 53 – Evento .blur

» **`.click`**

O evento `.click` é acionado quando algum elemento de um documento HTML é clicado. A Figura 54 apresenta um trecho do código do bate papo do CMS, esse trecho será executado quando um usuário clicar em um elemento com `id` igual a `ok`. A Figura 55 apresenta o elemento cujo evento de click dispara o evento mostrado na Figura 54. No bate papo quando o usuário preencher uma mensagem ele a enviará ao servidor através do evento `.click` do botão `"Enviar"`, esse evento processará o envio da mensagem.

```
$('#ok').click(function(){
    enviar();
});
```

Figura 54 – Evento .click

```
<input id="ok" type="button" value="Enviar">
```

Figura 55 – Botão de envio do bate papo

» **`.dblclick`**

O evento `.dblclick` é disparado quando um elemento é clicado duas vezes pelo usuário.

» **.change**

O evento .change é disparado quando um elemento muda de valor. Esse evento é muito utilizado em campos de formulário do tipo `select`. A Figura 56 apresenta um exemplo do uso do evento `.change` em um `select` que lista os estados brasileiros.

```
$('select').change(function() {
    alert("Você escolheu uma UF");
});
```

Figura 56 – Evento .change

» **.focus**

O evento `.focus` ocorre quando um elemento recebe foco, por exemplo, quando uma caixa de texto é selecionada. Na Figura 57 quando o usuário clica sobre um campo de formulário para digitar seu CPF, uma janela de alerta é exibida com informações.

```
$('input').focus(function() {
    alert("Use apenas números para informar seu CPF");
});
```

Figura 57 – Evento .focus

» **.keydown**

O evento `.keydown` é disparado quando o usuário pressiona uma tecla qualquer do teclado. A Figura 58 apresenta um exemplo de seu uso.

```
$('input').keydown(function() {
    alert("Uma tecla foi pressionada");
});
```

Figura 58 – Evento .keydown

» **.keyup**

O evento `.keyup` ocorre quando o usuário solta uma tecla do teclado que antes estava pressionada. Um exemplo de evento é apresentado na Figura 59.

```
$('input').keyup(function() {
    alert("Uma tecla deixou de ser pressionada");
});
```

Figura 59 – Evento .keyup

» **.load**

O evento `.load` é disparado quando o elemento ao qual ele se referencia termina de ser carregado na página HTML. Esse evento quando relacionado ao objeto `window` será disparado após o carregamento total da página. Na Figura 60 é exemplificado seu uso com o elemento `<body>` do HTML.

```
$('body').load(function() {
    alert("O corpo do HTML foi carregado");
});
```

Figura 60 – Evento .load

» **`.mouseover`**

O evento `.mouseover` ocorre quando o usuário passa o mouse por cima de um elemento. A Figura 61 exemplifica o uso desse evento..

```
$('input').mouseover(function() {
    alert("Passou o mouse por cima do elemento");
});
```

Figura 61– Evento .mouseover

» **`.mouseout`**

Quando o mouse sai de cima de um elemento o evento `.mouseout` é disparado. A Figura 62 a sintaxe do evento é apresentada.

```
$('input').mouseover(function() {
    alert("Passou o mouse por cima do elemento");
});
```

Figura 62 – Evento .mouseout

» **`.select`**

Quando o usuário seleciona um texto ou fragmento de texto de um elemento o evento `.select` é processado. Esse evento está disponível somente para elementos destinados à entrada de textos, como `input` e `textarea`. Na Figura 63 um exemplo da sintaxe do evento `.select` é mostrado..

```
$('input').select(function() {
    alert("Você selecionou um texto");
});
```

Figura 63 – Evento .select

» **`.submit`**

Quando um formulário é enviado o evento `.submit` é disparado. Ele é freqüentemente utilizado por programador para verificar ser processado quando um formulário é enviado para verificar se dados inseridos nos campos estão corretos. A Figura 64 exemplifica o uso do evento `.submit`.

```
$('input').submit(function() {
    alert("Dados do Formulário enviados");
});
```

Figura 64 – Evento .submit

Efeitos

Simplificar a implementação de efeitos é um dos objetivos do jQuery. O programador consegue enriquecer a experiência do usuário no uso de interfaces através de efeitos como esconder e revelar conteúdos ou um menu que mostra seus subitens de forma visualmente agradável através de suaves transições. A biblioteca jQuery permite implementar esses efeitos sem o detrimento da usabilidade da interface ou bloqueio do acesso de uma página. A seguir são apresentados alguns métodos para obtenção de efeitos em elementos HTML.

» **`.show`**

A sintaxe do método `.show` na sintaxe mostrada na Figura 65 revela o elemento associado a ele (elemento `<div>`) de maneira abrupta.

```
$('div').show();
```

Figura 65 – Método show sem parâmetro

O método `.show` pode possuir parâmetros. No código da Figura 66 o método `.show` apresenta o parâmetro `slow`, que indica que o elemento irá se revelar de forma lenta. Ao invés do parâmetro `slow` outros valores são possíveis tais como `normal` ou `fast` (velocidade normal e rápida respectivamente). A Figura 67 exemplifica o uso do método `.show` com um parâmetro numérico, esse parâmetro indica a velocidade em milissegundos com que o elemento irá aparecer na página.

```
$('div').show('slow');
```

Figura 66 – Revelação lenta do elemento DIV

```
$('div').show(3000);
```

Figura 67 – Revelação em 3 segundo do elemento DIV

O método show permite definir uma função que será executada quando o efeito terminar. Na Figura 68 assim que o elemento for revelado um alerta aparecerá.

```
$('div').show('normal', function(){
    alert("Elemento DIV revelado");
});
```

Figura 68 – Função com execução subseqüente ao efeito

» **`.hide`**

O efeito `.hide` tem um funcionamento contrário do método `.show`, mas possui sintaxe semelhante. O efeito `.hide` esconde o elemento associado a ele. A Figura 69 apresenta as sintaxes possíveis do método `.hide`.

```
$('input').hide();

$('table').hide('slow');

$('form').hide(5000);

$('div').hide('fast', function(){
    alert("Elemento DIV escondido");
});
```

Figura 69 – Sintaxes do método hide

» **`.toggle`**

Com o uso do método `.toggle` se um elemento está visível ele fica invisível, e vice versa. A Figura 70 apresenta a sintaxe do método.

```
$('div').toggle();
```

Figura 70 – Método toggle

» **`.slideDown`**

O método .slideDown se associado a um elemento escondido fará com que ele se revele suavemente. O método faz a transição do invisível para o visível por meio do aumento gradativo da altura do elemento. Os parâmetros possíveis são `slow`, `normal` ou `fast`. Na Figura 71 um exemplo de sintaxe é apresentado.

```
$('div').slideDown('fast');
```

Figura 71 – Método slideDown

» **`.fadeIn`**

O efeito `.fadeIn` é destinado para a criação de um efeito de revelar o elemento associado a ele que está escondido através do aumento gradativo da opacidade, saindo da opacidade 0 (invisível) para a opacidade 100% (visível). Os parâmetros possíveis são `slow`, `normal` ou `fast` ou um valor numérico que indica o tempo em milissegundos que durara o efeito. Uma função ainda poderá ser definida para ser executada assim que o elemento reaparecer. Na Figura 72 alguns exemplos de sintaxe são apresentados.

```
$('input').fadeIn('fast');

$('table').fadeIn(3000);

$('div').fadeIn('slow', function(){
    alert("O elemento DIV reapareceu gradativamente");
});
```

Figura 72 – Sintaxes do método fadeIn

» **`.fadeOut`**

O efeito `.fadeOut` tem funcionamento contrário ao criado pelo método `.fadeIn`, tendo sintaxe semelhante. A aplicação dessa função cria o efeito de esconder

um elemento através da diminuição gradativa da opacidade. A Figura 73 apresenta alguns exemplos de sintaxe.

```
$('input').fadeOut('slow');

$('table').fadeOut(5000);

$('div').fadeOut('fast', function(){
    alert("O elemento DIV desapareceu gradativamente");
});
```

Figura 73 – Sintaxes do método fadeOut

9.7. Notas Bibliográficas sobre a seção 9.6

Essa seção foi baseada no livro ***jQuery – A biblioteca do programador JavaScript***, um excelente livro para se aprofundar em jQuery, possui diversos exemplos e uma ampla documentação. O autor é o conceituado Maurício Samy Silva, que também desenvolve o site Maujor (*www.maujor.com*), referência na área de desenvolvimento para a web.

9.8. Ajax com jQuery

Introdução

AJAX é a sigla em inglês para *Asynchoronous JavaScript and* XML. Trata-se de uma técnica de carregamento de conteúdos em uma página Web com o uso de JavaScript e XML. O objeto `XMLHttpRequest` é o responsável pelo funcionamento do AJAX, que pode ser "traduzir" por requisição XML com o uso do protocolo HTTP. É uma técnica empregada para carregar conteúdo de um servidor (armazenada em um banco de dados ou não), sem necessidade de haver *reload*, ou seja sem necessidade de haver o recarregamento de toda a página para ocorrer a sua atualização (como ocorre com as aplicações Web tradicionais).

Ajax sem jQuery

Uma revisão do uso tradicional do Ajax é interessante para conhecer o seu mecanismo de funcionamento, pois no uso com jQuery algumas funções ficam transparentes ao desenvolvedor. O objeto `XMLHttpRequest` é o responsável pelo funcionamento do Ajax,e para a criação de uma instância desse objeto a sintaxe é como a apresentada na Figura 74:

```
var req = new XMLHttpRequest();
```

Figura 74 – Instanciando o objeto

A criação deste objeto varia de acordo com alguns navegadores. Caso o navegador seja o Internet Explorer nas versões 5 e 6 a instanciação do objeto se dá como mostrado na Figura 75.

```
var req = new ActiveXObject("Microsoft.XMLHTTP");
```

Figura 75 – Instanciando o objeto no Internet Explorer

Para a criação do objeto `XMLHttpRequest` nas versões mais antigas do Internet Explorer é utilizada a sintaxe `Microsoft.XMLHTTP`. Nas versões mais recentes a sintaxe `Msxml2.XMLHTTP` é empregada. Nas versões 7 e 8 do Internet Explorer o objeto `XMLHttpRequest` se tornou um objeto nativo do navegador, podendo ser instanciado como mostrado na Figura 74.

Instintivamente criar-se-ia um código que teste primeiramente o navegador para assim criar o objeto utilizando a sintaxe mais apropriada, porém este modo não é muito utilizado. A criação é feita através de tentativa e erro. Tenta-se criar o objeto, caso ocorra um erro é feita uma tentativa de criação por outra sintaxe e assim sucessivamente. A Figura 76 apresenta a criação do objeto XMLHttpRequest utilizando a sintaxe `try...catch`, ou seja, o que se segue a `try` é uma tentativa de executar algo, caso ocorra algum erro esse é "capturado" pelo `catch` e tratado.

```
1  function inicia_ajax(){
2      var req;
3
4      try {
5          req = new ActiveXObject("Microsoft.XMLHTTP");
6      } catch(e) {
7          try {
8              req = new ActiveXObject("Msxml2.XMLHTTP");
9          } catch(ex) {
10             try {
11                 req = new XMLHttpRequest();
12             } catch(exc) {
13                 alert("Esse browser não suporta Ajax");
14                 req = false;
15             }
16         }
17     }
18     return req;
19 }
```

Figura 76: Interface do Documento com marcações de apresentação

Na Tabela 13 acompanha-se o código da Figura 76 comentado.

Tabela 13: Código comentado

Linha(s)	Comentário
1	Criação da função que será sempre chamada para criar o objeto `XMLHttpRequest`
5	Cria o objeto se o navegador do usuário é o Internet Explorer 5 até 6
8	Cria o objeto se o navegador do usuário possui versões mais recentes do Internet Explorer
11	Sintaxe padrão para a criação do objeto `XMLHttpRequest` para os demais navegadores.
13 e 14	Caso o navegador não suporte Ajax uma caixa de alerta é apresentada.
18	A função retorna o objeto criado na variável `req`

Depois de criada a instância do objeto `XMLHttpRequest` o próximo passo é estabelecer a comunicação com o servidor, no qual são requisitadas informações ou arquivos. Os seguintes métodos cumprirão essa função:

- Ação disparadora de evento `onreadystatechange`
- Método `open`
- Método `send`
- Método `setRequestHeader`

» onreadystatechange

Uma ação disparadora de evento é uma ação iniciada por um usuário (como um evento `onclick` em um elemento da interface), mas não exclusiva dele, por exemplo, quando uma página é carregada um evento pode ser disparado através da sintaxe `window.onload`.

A ação `onreadystatechange` é disparada pelo servidor quando ele envia para o cliente uma informação de que alguma atualização ou mudança ocorreu nele. A ação é executada sempre que ocorre uma mudança no valor da propriedade `readyState` do objeto `XMLHttpRequest` A propriedade readyState pode assumir cinco valores, como mostrado na Tabela 14:

Tabela 14: Valores da propriedade readyState

Valor	Status/Interpretação
0	`UNSENT/` Comunicação não iniciada
1	`OPENED/` Início da comunicação. A comunicação foi iniciada sem envio de dados.
2	`HEADERS RECEIVED/` Carregamento finalizado
3	`LOADING/` Servidor em processo de envio da resposta à requisição.
4	`DONE/` Servidor acaba de enviar a resposta à requisição.

A Figura 77 apresenta a implementação da ação disparadora do servidor.

```
1  var req = inicia_ajax();
2  if(req){
3      req.onreadystatechange=function(){
4          if(req.readyState == 4){
5              if(req.status == 200){
6                  document.getElementById("chat").innerHTML = req.responseText;
7              }
8          }
9      };
10 }
```

Figura 77 – Ação disparadora do servidor do bate papo

Comentários a respeito do código da Figura 77 são mostrados na Tabela 15..

Tabela 15 – Figura 77 comentada

Linha(s)	Comentário
1	Iniciamos o objeto `XMLHttpRequest`, que será referenciado por `req`.
2	Testamos se o objeto foi criado.
3	`onreadystatechange` definirá a função que será chamada quando o servidor alterar a propriedade `readyState` do objeto `req`
4	Se a propriedade `readyState` indicar que certa ação está com o status `DONE`, ou seja, o servidor enviou a resposta a alguma requisição efetuada, ele executará as linhas 5 a 7.
5	O servidor envia um cabeçalho HTTP alterando a propriedade status com um valor numérico de 3 dígitos. Quando o valor da propriedade status é 200 indica que a requisição efetuada foi bem sucedida.
6	A propriedade `responseText` retorna os dados enviados pelo servidor como resposta à requisição, foi feita (em outra parte do código não presente na Figura 77) uma requisição pelas últimas 30 mensagens do bate papo, a propriedade `responseText` conterá o HTML das mensagens enviadas pelo servidor ao navegador. Este HTML enviado será colocado no elemento `<div id="chat">` que criamos.

» **Método open**

O método `open` é utilizado para informar ao servidor qual o endereço do arquivo ao qual ele requisita. A Figura 78 mostra a sintaxe do método que foi utilizado no bate papo.

```
req.open('GET', "msg.php", true);
```

Figura 78 – Sintaxe do método open

O primeiro parâmetro do método `open` indica que método de envio dos dados pode ocorrer por `GET` ou `POST`. Na Figura 78 é utilizado o `GET`. O segundo parâmetro indica a `url` do arquivo que se deseja obter, e o terceiro é um parâmetro facultativo, que quando está com o valor `true` indica que a comunicação com o servidor é assíncrona.

» **Método send**

O método `send` inicia a requisição que foi definida pelo método `open`. Ela admite somente um parâmetro que pode ser um conjunto de dados que se deseja enviar ao servidor ou o valor `null` que significa que a requisição não envia dados (como na Figura 79).

```
req.send(null);
```

Figura 79 – Sintaxe do método send

A Figura 80 apresenta o código da interface do bate papo. A apresentação da interface do bate papo no navegador é mostrada na Figura 81.

```
1  <html>
2      <head>
3          <title>Bate Papo Naturalista</title>
4          <script type="text/javascript" src="ajax.js" ></script>
5          <link rel="stylesheet" href="estilo.css" />
6      </head>
7      <body>
8          <div id="pagina_chat">
9              <div id="chat"> </div>
10                 <form  id="chat">
11                     <input id="msg" type="text" size="20">
12                     <input id="ok" type="button" value="Enviar" onClick="enviar()">
13                 </form>
14             </div>
15         </div>
16     </body>
17 </html>
```

80 – Código HTML da interface do chat

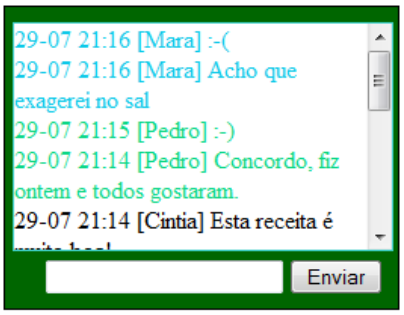


Figura 81 – Interface do bate papo

O comportamento Ajax é apresentado na Figura 82.

```
1  function inicia_ajax(){
2      /* ... criação do objeto XMLHttpRequest ... */
3  }
4
5  function enviar() {
6      var req = inicia_ajax();
7      if(req){
8          req.onreadystatechange= function(){ /* implementação da função disparadora */ };
9          var mensagem = document.getElementById("msg").value;
10         var url = "msg.php?msg=" + escape(mensagem);
11         req.open('GET',url, true);
12         req.send(null);
13     }
14     document.getElementById("msg").value = "";
15 }
16
17 function ler() {
18     var tempoChat = 2000;
19     var req = inicia_ajax();
20     if(req){
21         req.onreadystatechange=function(){
22             /* implementação da função disparadora */
23             setTimeout("ler()", tempoChat);
24         };
25         req.open('GET', "msg.php?", true);
26         req.send(null);
27     }
28 }
29
30 window.onload = ler();
```

Figura 82 – Implementação do comportamento Ajax do bate papo

Na Tabela 15 acompanha-se o código da Figura 82 comentado.

Tabela 15 – Código da Figura 82 comentado

Linha(s)	Comentário
1 a 3	Cria o objeto Ajax
5 a 15	Quando um usuário envia uma mensagem, a função `enviar()` é chamada. Ela inicia o objeto Ajax, define a função disparadora do servidor (`onreadystatechange`), captura a mensagem digitada (linha 9), prepara a url que será requerida no servidor (linha 10), informa ao servidor qual o arquivo e em seguida efetua a requisição (linhas 11 e 12). Ao final limpa o campo de mensagem (linha 14).
17 a 28	A função `ler()` é chamada de 2 em 2 segundos, ela é responsável por buscar no servidor as mensagens. Na linha 23 a sintaxe `setTimeout` chamará a função `ler()` no tempo definido. Em seguida definimos novamente o arquivo que será requisitado e executamos a requisição (linhas 25 e 26).
30	Quando o documento terminar de ser carregado no navegador do usuário a função `ler()` é chamada, é nele que o script inicia, sem interferência do usuário.

A página *msg.php* que é chamada nos exemplos do bate papo desse trabalho é apresentada na Figura 83.

```
1  <?php
2      $conexao = mysql_connect("endereco","usuario", "senha").
3      $db = mysql_select_db("bd");
4
5      if($_GET["msg"]){
6          $data = time();
7          $usuario = $_SESSION["nome"];
8          $mensagem = $_GET["msg"];
9
10         $sql = "INSERT INTO batepapo (data, usuario, mensagem)
11                 VALUES '$data', '$usuario', '$mensagem')";
12         $resultado = mysql_query($sql) or die (mysql_error());
13     }
14
15     $sql = "SELECT * FROM batepapo ORDER BY id DESC LIMIT 20";
16     $resultado = mysql_query($sql) or die (mysql_error());
17
18     while ($linha_bp = mysql_fetch_array($resultado)){
19         $data = date('d-m H:i',$linha["data"]);
20         $usuario = $linha["usuario"];
21         $mensagem = $linha["mensagem"];
22
23         echo "$data [$usuario] $mensagem <br />";
24     }
25 ?>
```

Figura 83 – msg.php, arquivo que grava e imprime as mensagens

Na Tabela 16 acompanha-se o código da Figura 83 comentado.

Tabela 16 – Código da Figura 83 comentado

Linha(s)	Comentário
2 e 3	Conexão ao banco de dados
5 a 13	Caso o usuário tenha enviado uma mensagem a assertiva `$_GET[“msg”]` será verdadeira e o código será executado. A data e hora do envio será armazenada na variável `data` (linha 6), o nome do usuário é capturado da variável de sessão (que é global enquanto o usuário estiver logado no site, linha 7) e a variável `mensagem` armazena o que foi digitado pelo usuário (linha 8). A `data`, `usuário` e `mensagem` são armazenados na tabela `bate-papo` (linhas 10, 11 e 12)
10 e 11	São consultadas as últimas 20 mensagens armazenadas.
18 a 24	O resultado da consulta é percorrido e impresso.

Ajax com jQuery

Na seção 9.6 é visto que jQuery facilita a implementação do código JavaScript, nessa seção veremos a estrutura que o jQuery oferece para implementar aplicações Ajax.

Funções de requisição

A seguir serão apresentadas as funções jQuery para realizar requisições Ajax, serão vistos sua sintaxe, aplicações e funcionamento. O coração do funcionamento do Ajax é o objeto `XMLHttpRequest` mas no jQuery a instanciação deste objeto é transparente para o programador, além de serem transparentes também alguns métodos que o Ajax utiliza na programação tradicional.

» **$.get**

A função $.get é uma requisição HTTP com o uso do método GET. Ela admite somente uma função que será chamada ao final da requisição. Se forem necessárias funções para tratamento de erros então é feito o uso da função $.ajax. A Figura 84 e Figura 85 apresentam a requisição HTTP utilizada no bate papo.

```
1  $.get('msg.php',
2        function(dado){ $('#chat').html(dado); },
3        'html');
```

Figura 84– Função $.get

A função $.get admite 4 parâmetros, na Figura 84 temos 3, e no código da Figura 85 temos todos os 4 parâmetros, sendo que o primeiro é sempre obrigatório. O código da Figura 84 é utilizado no exemplo do CMS do Portal de Receitas para a consulta das últimas mensagens adicionadas no banco de dados. O código da Figura 85 envia a mensagem que o usuário digita no bate papo, a armazena no banco de dados e por fim retorna as últimas mensagens dos usuários. A Tabela 17 apresenta o código da Figura 84 comentado.

Tabela 17 – Código da Figura 84 comentado

Linha(s)	Comentário
1	Função de requisição Ajax $.get possui no exemplo 3 parâmetros, o arquivo a ser requisitado, uma função para ser executada ao final da requisição e o tipo do dado que está transitando. O primeiro parâmetro, o url, é obrigatória na função $.get e está definida na linha 1. O URL do arquivo requisitado, no caso do exemplo é uma página com o resultado em HTML de uma consulta ao banco de dados das últimas mensagens enviadas pelos usuários.
2	Definição da função de retorno. A variável dado que é o parâmetro da função, recebe o conteúdo de retorno da requisição, no caso o html com as últimas mensagens do bate papo, que está sendo colocado no elemento DIV através da sintaxe $('#chat').html(dado).
3	Tipo de dado que está transitando na requisição, os valores possíveis para este parâmetro são: "XML", "html", "script", "json" ou "text"

Na Figura 85 o código da função $.get possui 4 parâmetros. O segundo parâmetro é um conjunto de pares variável/valor que é utilizado para enviar dados ao servidor..

```
1  var mensagem = $('#msg').val();
2
3  $.get('msg.php',
4        {msg: mensagem},
5        function(dado){ $('#chat').html(dado); },
6        'html');
```

Figura 85 – Função $.get enviando dado para o arquivo

Na Tabela 18 acompanhamos o código da Figura 85 comentado. Na Figura 86 temos o código que determina o comportamento do nosso bate papo.

Tabela 18 – Código da Figura 85 comentado

Linha(s)	Comentário
1	A variável mensagem captura o valor que o elemento de `id` igual a `msg` possui, no caso este elemento é um campo de texto no qual o usuário do bate papo digita a mensagem.
2	Definição da função de retorno. A variável `dado` que é o parâmetro da função, recebe o conteúdo de retorno da requisição, no caso o html com as últimas mensagens do bate papo, que está sendo colocado no elemento DIV através da sintaxe `$('#chat').html(dado)`.
3	Tipo de dado que está transitando na requisição, os valores possíveis para este parâmetro são: "`XML`", "`html`", "`script`", "`json`" ou "`text`"

```
1  $(document).ready(function(){
2      ler();
3      $('#msg').focus();
4      $('#ok').click(function(){
5          enviar();
6      });
7  });
8
9  var ler = function() {
10     $.get('msg.php',
11           function(dado){ $('#chat').html(dado); },
12           'html');
13     setTimeout('ler()', 2000);
14 }
15
16 var enviar = function() {
17     var mensagem = $('#msg').val();
18     $.get('msg.php',
19           {msg: mensagem},
20           function(dado){ $('#chat').html(dado); },
21           'html');
22     $('#msg').val('');
23     $('#msg').focus();
24 }
```

Figura 86 – Implementação do comportamento Ajax com jQuery do bate papo

» **`$.post`**

A função `$.post` é uma requisição HTTP com o uso do método `POST`. A sintaxe da função é similar a da função `$.get`, onde consta o termo `get` substituímos por `post`.

» **`$.getScript`**

A função `$.getScript` é utilizada para realizar requisições de arquivos JavaScript que se encontram em um domínio remoto (domínio distinto da página que faz a requisição).

A Figura 87 apresenta um documento HTML com um elemento `div`, que será estilizado através do script importado pela função `$.getScript`. O arquivo JavaScript para a qual a função `.getScript` faz requisição é um plugin que arredonda as bordas de alguns elementos HTML.

```
1  $.getScript('http://github.com/malsup/corner/raw/master/jquery.corner.js',
2              function() {
3                  $('div').corner();
4              });
```

Figura 87 – Função $.getScript

Na Tabela 19 são mostrados os comentários a respeito do código da Figura 87.

Tabela 19 – Código da Figura 87 comentado

Linha(s)	Comentário
1	O primeiro parâmetro é a requisição do arquivo JavaScript remoto. O arquivo é o *jquery.corner.js* hospedado em *http://github.com/malsup/corner/raw/master/jquery.corner.js*. É um plugin para estilização de cantos cuja documentação encontra-se em *http://jquery.malsup.com/corner/*.
2	Definição da função de retorno, que arredondará todos os elementos `div` da página.

» **`.load`**

O método `.load` é uma função jQuery simplificada que é utilizada para realizar requisições HTTP com o uso do método `GET` por padrão. Caso sejam passados dados pelo método a requisição assumirá um envio HTTP pelo método `POST`. Na Figura 88 há um exemplo de sintaxe do método `.load` que possui a mesma função da função `$.get` apresentada na Figura 84.

```
$('#chat').load('msg.php');
```

Figura 88 – Método .load

» **`$.ajax`**

A função `$.ajax` carrega dados do servidor utilizando uma requisição HTTP. As funções vistas anteriormente também podem ser feitas com essa função, porém pode ela pode ser configurada com muito mais parâmetros que aquelas.

A função `$.ajax` admite um parâmetro. Esse parâmetro é um objeto constituído de um conjunto de pares chave/valor que serão usados para iniciar e manipular a requisição.

Na tabela 20 temos algumas das possíveis chaves/valores do objeto que é o parâmetro da função `$.ajax`. No exemplo da Figura 89 utilizam-se algumas chaves, tais como `url`, `dataType`, `data`, `type`, `success` e `error`.

Tabela 20 – Algumas chave/valor que constituem o objeto da função $.ajax

Chave	Valor
`cache`	Default `true` que evita que a página requisitada vá para o cachê do navegador.
`complete`	Função chamada quando a requisição termina. Possui dois argumentos, o objeto `XMLHttpRequest` e uma string que define o status de como terminou a requisição.
`contentType`	String que descreve o tipo de conteúdo que está sendo enviado ao servidor, por exemplo "`text/xml`".
`data`	Os dados que serão enviados ao servidor.
`dataType`	O tipo de dado que se espera receber do servidor, os tipos possíveis são: `XML`, `html`, `script`, `json`, `jsonp` e `text`. Este tipo deve ser definido corretamente, caso contrário não será possível manipular os dados retornados.
`error`	Função chamada quando ocorre um erro na requisição. Possui dois argumentos, o objeto `XMLHttpRequest` e uma string que descreve o tipo de erro que ocorreu.
`sucess`	Função que é chamada quando a requisição foi executada com sucesso. Possui dois argumentos, os dados retornados e uma string descrevendo o status.
`type`	O valor default é `GET` e define o tipo de requisição. Os tipos normais são `GET` e `POST`, porém outros são possíveis como `PUT` ou `DELETE`.
`url`	O arquivo para qual a requisição é feita.

Na Figura 89 há um exemplo de sintaxe para a função `$.ajax` que realiza a mesma função do código da Figura 84, porém há o acréscimo de uma função que será requisitada caso ocorra qualquer erro durante a requisição.

```
$.ajax({
    url: 'msg.php',
    dataType: 'html',
    data: {msg: mensagem},
    type: 'GET',
    success: function(dado, statusReq){
        $('#chat').html(dado);
    },
    error: function(xhr, tipo){
        $('#chat').html('Erro: ' + xhr.status + ' - '
        + xhr.statusText + ' <br /> Tipo de erro: ' + tipo);
    }
});
```

Figura 89 – Função $.ajax

Requisições XML

A biblioteca jQuery oferece suporte para extrair conteúdo de um arquivo XML. A Figura 90 apresenta o arquivo *mousse.xml*, que armazena os dados de uma receita de cozinha de nosso CMS.

```
<?xml version="1.0" encoding="utf-8"?>
<receita>
    <titulo> Mousse de Maracujá </titulo>
    <imagem> imagens/mousse_maracuja.jpg</imagem>
    <ingredientes>
        <![CDATA[
        <ul>
            <li> 2 latas de leite condensado </li>
            <li> 4 maracujás </li>
        </ul>
        ]]>
    </ingredientes>
    <preparo>
        <![CDATA[
        <p>Coloque o leite condensado em uma vasilha e reserve.</p>
        <p>Retire a polpa dos maracujás e com a ajuda de uma peneira retire o suco.</p>
        <p>Junte o leite condensado e o suco do maracujá. Misture até ficar homogêneo.</p>
        <p>Leve a geladeira. Enfeite com sementes de maracujá, cerejas e folhas de hortelã.</p>
        ]]>
    </preparo>
</receita>
```

Figura 90 – Arquivo XML

Ao se extrair as informações do documento da Figura 90, serão apresentados os conteúdos a respeito de uma receita com título, imagem de apresentação, ingredientes e modo de preparo. A Figura 91 apresenta o código Ajax que extrai o conteúdo do arquivo XML e o imprime.

```
$.ajax({
    url: 'receita.xml',
    dataType: 'xml',
    type: 'POST',
    success: function(dado, textStatus){
        var html = '<h1>' + $(dado).find('titulo').text(); + '</h1>';
        html += '<p><img src="' + $(dado).find('imagem').text() + '" /> </p>';
        html += '<h2> Ingredientes </h2>';
        html += '<p>' + $(dado).find('ingredientes').text() + '</p>';
        html += '<h2> Modo de Preparo </h2>';
        html += '<p>' + $(dado).find('preparo').text() + '</p>';
        $('#rec').html(html);
    },
    error: function(xhr, erro){
        $('#rec').html(xhr.statusText + erro);
    }
});
```

Figura 91 - Requisição XML

Na Tabela 21 os comentários a respeito do código da Figura 91.

Tabela 21 – Código da Figura 91 comentado

Linha(s)	Comentário
2	Definimos o arquivo XML
3	Informamos que o tipo de dado que o servidor retornará será do tipo XML
4 a 13	Definição da função que será executado caso a requisição seja concluída com sucesso. A variável `dado` contém os dados enviados pelo servidor. A variável `html` conterá os dados que estão na estrutura XML formatados com marcações HTML que será exibido em um `<div>` com `id` igual a `rec` (linha 12 da Figura). A sintaxe `$('dado').find('titulo').text()` define o seguinte: percorra o conjunto de elementos de `$('dado')`, encontre os elementos `titulo` e deles extraia seus textos. Como o elemento `titulo` é único o resultado é um texto somente.
14 a 16	Função que será chamada quando ocorrer um erro na requisição;

Acessibilidade Ajax

Um dos grandes problemas no uso do Ajax é acessibilidade. Muitos navegadores suportam JavaScript, mas algumas tecnologias portáteis e navegadores somente textuais não suportam. O W3C (*World Wide Web Consortion*) diz no WCAG 1.0 (*Web Content Accessibility Guidelines* 1.0) que um site deve ser navegável mesmo quando o JavaScript do navegador estiver desativado.

Alguns desenvolvedores tem por prática utilizar requisições Ajax em todo site: em links, em menus etc. Mas isso não é necessário e também torna o site inacessível principalmente para os motores de buscas, pois esses sistemas não indexam dados que não estejam no código fonte do site. Dados requisitados através de requisições Ajax não são impressas no código fonte de páginas HTML.

É importante usar Ajax com moderação, e respeitando a acessibilidade. Um site que possui elementos Ajax deve funcionar em navegadores sem suporte a JavaScript, então ao desenvolver um site deve-se programar inicialmente sem JavaScript. Depois de concluído funcionalidades podem ser acrescentadas, porém, sempre respeitando questões como usabilidade e acessibilidade.

9.8. Notas Bibliográficas sobre a seção 9.8

A seção 9.8 se baseou no livro ***Ajax com jQuery – Requisições Ajax com a simplicidade de jQuery***, outro excelente livro que possui diversos exemplos e uma ampla documentação. O autor é também o conceituado Maurício Samy Silva, que também desenvolve o site Maujor (*www.maujor.com*), referência na área de desenvolvimento para a web.

Em *http://msdn.microsoft.com/pt-br/library/cc534581%28VS.85%29.aspx* é descrito algumas novas propriedades do Ajax criadas pela Microsoft no Internet Explorer 8, onde há comentários a respeito do objeto `XMLHttpRequest` já estar nativo nos navegadores Internet Explorer 7 e 8. Nesse endereço está sendo descrito algumas novas funções que estão implementadas no objeto, com novas possibilidades.

Existem muitos sites na web que possuem artigos sobre a acessibilidade em sites que utilizam Ajax, neste trabalho foram utilizados os sites *http://juliogreff.net* e *http://www.designacessivel.net.*

Referências Bibliográficas

Anderson, P. (2008). What is Web 2.0? - Ideas, technologies and implications for education. Disponível em <*www.jisc.ac.uk/techwatch*>. Acesso em: 10/04/2010.

Berners-Lee, T. (1999). "Weaving the Web: the past, present and future of the World Wide Web by its inventor". London : Orion Business.

Coelho, D. V. (2009). "Drupal x Joomla x Wordpress, qual o melhor CMS". Disponível em: <*http://www.portal2web.com.br/software-livre/drupal-x-joomla-x-wordpress-qual-o-melhor-cms.html*>. Acesso em: 15/07/2010.

Dall'Oglio, P. (2007). "PHP: Programando com Orientação a Objetos". Editora Novatec, São Paulo.

Design Acessível. (2007). "Acessibilidade com Ajax". Disponível em <*http://www.designacessivel.net/wiki/acessibilidade-com-ajax*>. Acesso em: 01/08/2010

Greff, J. (2006). "Ajax vs. Acessibilidade". Disponível em: <*http://juliogreff.net/ajax-vs-acessibilidade*>. Acesso em: 02/08/2010.

Kristaly, D. M., Sisak F., Truican, I., Moraru, S., Sandu, F. (2001). "Web 2.0 technologies in web application development".

Kostur, P. (2006). "Incorporating Usability into Content Management".

Microsoft. (2010). "Aprimoramentos de XMLHttpRequest no Internet Explorer 8". Disponível em <*http://msdn.microsoft.com/pt-br/library/cc534581%28VS.85%29.aspx*>. Acesso em 29/07/2010.

Montalvo, C. (2008). "Comezando MVC com php". Disponível em: <*http://www.calinsoft.com/2008/08/comenzando-mvc-con-php.html*>. Acesso em: 28/07/2010.

Nemetral. (2008). "A gentle introduction to MVC". Disponível em: <*http://nemetral.net/2008/07/31/a-gentle-introduction-to-mvc-part-1/*>. Acesso em: 10/07/2010.

O' Reilly, T. (2007). "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software". Communications & Strategies. Disponível em <*http://ssrn.com/abstract=1008839*>. Acesso em 01/4/2010.

Pereira, J. C. L., Bax, M. P. (2002). "Introdução à Gestão de Conteúdos".

Silva, M. S. (2008). "jQuery: A Biblioteca do Programador JavaScript", Editora Novatec, São Paulo.

Silva, M. S. (2009). "Ajax com jQuery: Requisições Ajax com a simplicidade de jQuery", Editora Novatec, São Paulo.