UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

TÉCNICAS DE ENGENHARIA DE SOFTWARE PARA PROJETOS DE APLICAÇÕES WEB

TRABALHO DE GRADUAÇÃO

Guilherme Lorenzoni Algarve

Santa Maria, RS, Brasil
2008

TÉCNICAS DE ENGENHARIA DE SOFTWARE PARA PROJETOS DE APLICAÇÕES WEB

por

Guilherme Lorenzoni Algarve

Trabalho de Graduação apresentado ao Curso de Ciência da Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação.**

Profª. Msc. Oni Reasilvia de Almeida Oliveira Sichonany

Trabalho de Graduação N° 262

Santa Maria, RS, Brasil

2008

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada, aprova o Trabalho de Graduação

TÉCNICAS DE ENGENHARIA DE SOFTWARE PARA PROJETOS DE APLICAÇÕES WEB

elaborado por
Guilherme Lorenzoni Algarve

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação.

COMISSÃO EXAMINADORA:

Oni Reasilvia de Almeida Oliveira Sichonany, Msc.
(Presidente/Orientador)

Gedson Mário Borges Dal forno, Msc. (UFSM)

Iria Brucker Roggia, Msc. (UFSM)

Santa Maria, 28 de Julho de 2008.

AGRADECIMENTOS

Antes de tudo, agradeço a Deus por ter me dado coragem para lutar e sabedoria nas decisões da vida, pois através da fé Nele pude acreditar em mim mesmo.

À professora Oni, que me apoiou e orientou durante o percurso desse trabalho apesar das dificuldades e de sua vida atarefada, tratando o assunto como se a conclusão do trabalho fosse tão importante para ela quanto o é para mim.

À minha namorada Maristela, que não apenas foi minha estrela guia como também os "puxões de orelha" que precisava para voltar ao caminho quando desviado.

Ao meu filho Davi, cuja alegria e fé no pai me motivaram a continuar adiante apesar dos obstáculos da vida.

À minha mãe Sonia, que me supriu com todas as condições para chegar até onde estou, e pela fé e orgulho que ela tem em mim.

Ao meu pai Asdrúbal, cuja presença em espírito me guiou e exemplo serviu como inspiração para lutar.

Finalmente, agradeço ao meu amigo Tiago por ter me aconselhado através de sua experiência a seguir os passos corretos a caminho da vitória.

RESUMO

Trabalho de Graduação
Ciência da Computação
Universidade Federal de Santa Maria

TÉCNICAS DE ENGENHARIA DE SOFTWARE PARA PROJETOS DE APLICAÇÕES WEB

Autor: Guilherme Lorenzoni Algarve
Orientador: Msc. Oni Reasilvia de Almeida Oliveira Sichonany
Data e Local da Defesa: Santa Maria, 28 de Julho de 2008.

A tecnologia se desenvolve em ritmo acelerado, em especial no campo da informática. Tecnologias de informação relacionadas à internet tiveram uma notável evolução nos últimos anos, tendo atualmente um elevado grau de complexidade e importância na vida das pessoas, porém, técnicas e metodologias para a concepção e manuseio dos sistemas usados na grande rede mundial não acompanharam esse processo. Cientes desse fato, pesquisadores e desenvolvedores da área voltaram-se à criação de métodos formais e eficientes para o desenvolvimento dos hoje complexos e onipresentes sistemas *Web*.

O presente trabalho aborda o tema da sistematização do processo de criação de aplicações para internet introduzindo a história da evolução das páginas eletrônicas, apresentando características dos principais métodos usados fazendo um paralelo com métodos de engenharia de softwares convencionais. Será também apresentado um modelo para a criação de uma aplicação web para gerenciamento de bibliotecas usando um dos métodos apresentados, o UWE.

Palavras-chave: Engenharia Web; Aplicação Web; Engenharia de Software.

ABSTRACT

Undergraduation Final Work
Computer Science
Federal University of Santa Maria

WEB ENGINEERING TECHNIQUES FOR WEB APPLICATION PROJECTS

Author: Guilherme Lorenzoni Algarve
Advisor: Msc. Oni Reasilvia de Almeida Oliveira Sichonany

The technology evolves at fast pace, especially in the informatics field. Internet related information technologies have a notable evolution in the last years, actually having a high grade of complexity and importance in people lives, but, techniques and methodologies for creation and management of systems used in the world wide web not follow this process. Aware of this fact, field researchers and developers turn back to the creation of formal and efficient development methods of the today complexes andomnipresent Web systems.

This work approaches the process systematization theme of the internet applications creation process introducing the electronic pages evolution history, presenting the characteristics of the main methods used making a conventional software engineering comparison. Will be present to a model for a library management web system creation using one of the methods presented, the UWE.

Keywords: Web Engineering; Web Application; Software Engineering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

B2B	*Business to Business*
CGI	*Common Gateway Interface*
HTML	*Hypertext Markup Language*
OMG	*Object Management Group*
UML	*Unified Modeling Language*
UP	*Unified Process*
URL	*Uniform Resource Locator*
UWE	*UML-based Web Engineering*

SUMÁRIO

1 INTRODUÇÃO

De acordo com a história da civilização, os avanços científicos aceleram com o passar do tempo, e como pode ser visto nas últimas décadas, essa característica é bem evidente nas áreas tecnológicas, com destaque para tecnologias de informação.

Quando uma nova tecnologia é criada ou melhorada, outras relacionadas devem ser também criadas ou modificadas. Porém, por diversas razões, nem sempre isso acontece, ocasionando aplicações indevidas ou pouco proveitosas dessas tecnologias.

No caso das tecnologias de informação, essa corrida tecnológica é ainda mais acirrada, como é visto, por exemplo, na evolução de metodologias para desenvolvimento de *softwares*. Os primeiros programas para computador criados eram simples o suficiente para serem executados em hardwares disponíveis na época. Quando a capacidade dos computadores aumentou, foi permitida também a criação de programas mais úteis e complexos, envolvendo muitos desenvolvedores e maiores esforços na manutenção. Então se viu a necessidade do uso de metodologias para o desenvolvimento de *softwares*, nascendo assim a Engenharia de *Software*.

A área de tecnologias para *web* está passando por uma fase parecida com a que os programas convencionais passaram (KAPPEL, 2003), em que a complexidade estrutural e manutenção das páginas de *internet* estão além dos métodos usados para criação e gerenciamento. Atendendo a essa carência de sistematização, pesquisadores e desenvolvedores propuseram metodologias baseadas na engenharia de *software*, adaptadas para aplicações *web*: a Engenharia *Web*.

Nesse trabalho serão mostradas características das páginas *web* no decorrer da história da *internet*, como graus de complexidade e finalidades. Será abordada também a relevância da concepção sistematizada através da exposição das formas com que eram criadas as páginas de *internet*.

Será apresentada nesse trabalho em detalhes uma das técnicas de engenharia *web*, a UWE. Trata-se de uma metodologia que abrange todos os ciclos de vida da aplicação e tem como uma das principais características a extensão da linguagem de modelagem UML como adaptação para geração de diagramas e especificações próprias de aplicações feitas para *internet*.

Para evidenciar um exemplo prático da aplicação da engenharia *web* para criação de um sistema de informação, será apresentado um projeto de uma aplicação *web* de um sistema de biblioteca utilizando a abordagem UWE.

Finalmente, serão expostos os argumentos conclusivos sobre o tema, assim como a estimativa do impacto das técnicas de engenharia *web* sobre projetos futuros e os esforços feitos por pesquisadores e desenvolvedores para desenvolvimento de métodos.

2 REVISÃO DA LITERATURA

Este capítulo visa apresentar as motivações para a proposta da engenharia *web*, as características principais e os tipos de aplicações *web*. Será feito também um paralelo entre a engenharia de software convencional e a engenharia *web*, expondo suas características comuns, próprias e adaptações. Também serão introduzidas metodologias de engenharia *web* conhecidas com suas características principais como notação, paradigmas e pontos fortes.

2.1 Aplicações *web*

Nessa seção será apresentada a história das aplicações *web*, relacionando elementos e as necessidades que colaboraram com a criação da mesma. Serão também mostradas as categorias das aplicações *web* que existem e suas principais aplicações. Em seguida, serão definidas as características comuns em aplicações gerais e também características únicas de aplicações *web*.

2.1.1 História das aplicações *web*

Com a criação do computador, foi possível resolver problemas antes impossíveis ou inviáveis de serem solucionados. Tanto como dispositivos para armazenamento de grandes volumes de dados como solucionador de cálculos gigantescos e complexos, através de instruções dadas por humanos, muitas tarefas úteis podem ser feitas. Hoje em dia, o *software* está presente, explicitamente ou mesmo sem se fazer notar, em todos os aspectos de nossa vida, inclusive nos sistemas críticos que afetam a nossa saúde e o nosso bem-estar (PFLEEGER, 2004). Porém, à medida que a capacidade do *hardware* dos computadores e a complexidade dos problemas aumentavam, também aumentava o tempo para criar o conjunto de instruções ou programas e o número de pessoas necessárias para escrevê-las. Então foram criados métodos sistemáticos para auxiliar o projeto e manutenção desses programas, métodos esses definidos na Engenharia de *Software*. Através da engenharia de *software*, tornou-se viável criar e manusear aplicações com grande número de linhas de código com o mínimo de depuração e envolver vários programadores em um mesmo projeto com o mínimo de conflito.

O seguinte grande passo da evolução da tecnologia da informação consiste numa forma de computadores do mundo todo compartilhar informações e se comunicarem a grandes distâncias, a *Internet*. Em princípio, a grande rede era usada essencialmente para transmissão de informações, através de páginas simples contendo apenas textos e um código de marcação para formatação. À medida que evoluíam o *hardware* e tecnologias de redes e comunicação, essas

páginas se aperfeiçoaram com a adição de imagens, formulários, métodos de envios de dados, até se tornarem complexas aplicações. A *internet* tornou-se uma verdadeira plataforma para aplicações, propiciando assim condições para o nascimento de um novo conceito na tecnologia da informação: as Aplicações *Web*.

2.1.2 Tipos de aplicações *web*

Como mencionado anteriormente, desde a invenção da *internet* até os dias de hoje, as páginas, inicialmente usadas como simples meio de compartilhamento de texto, passaram por vários estágios evolutivos até se tornarem complexas aplicações, definindo assim categorias de aplicações *web* (MURUGESAN, 2005). As principais categorias de aplicações *web*, em ordem crescente de complexidade, são: páginas informativas, transacionais, de fluxo de trabalho, colaborativas e sociais, e serão apresentadas nas subseções a seguir.

2.1.2.1 Páginas informativas

As páginas informativas são documentos formatados em linguagem de marcação, utilizadas como meios de informação e não tinham meios de interação com o usuário. São muito usadas para divulgação de notícias, catálogos virtuais, livros *on-line* entre outros.

2.1.2.2 Páginas interativas

Com a introdução da tecnologia do portal de interface comum ou CGI, que é forma de comunicação que um servidor *web* emprega para enviar informações entre o navegador e um programa de computador em um servidor *web* (WEINMAN, 1997), e mecanismos para elementos dinâmicos dos navegadores, páginas interativas puderam ser criadas. Essas páginas permitem comunicação com o servidor através de formulários contendo botões, caixas de seleção e também meios de modificar a página de acordo com entradas do usuário. As principais aplicações para páginas interativas são formulários de registro, jogos *on-line* e personalização da apresentação de informações.

2.1.2.3 Sistemas transacionais

As páginas transacionais permitem que o usuário não apenas interaja através da leitura de informações, mas também permitem modificar informações do servidor através de transações. Esse mecanismo foi possível graças à introdução de sistemas de banco de dados nos servidores, que permitem formas de usuários modificarem e interagirem com dados comuns. Entre as aplicações comuns para sistemas transacionais estão sistemas de compras, bancários e de reservas de passagens *on-line*.

2.1.2.4 Sistemas baseadas em fluxo de trabalho

O estágio seguinte na evolução da capacidade de aplicações *web*, permite que sejam manuseados fluxos de trabalho entre companhias, empresas e autoridades públicas. Isso devido à introdução da tecnologia de *web services*, que permite interoperabilidade entre sistemas, ou seja, computadores e sistemas diferentes podiam se comunicar através de protocolos comuns. Exemplos típicos de aplicação são os sistemas B2B, usados para transações comerciais eletrônicas entre parceiros de negócios, e de administração pública, entre outros.

2.1.2.5 Sistemas colaborativos

Sistemas colaborativos têm como função oferecer um ambiente comum de trabalho e métodos de coordenação das interações dos usuários de um grupo. Através de um sistema colaborativo as pessoas podiam trabalhar em documentos comuns e participar reuniões virtuais mediadas pelo sistema. Exemplos desses sistemas são as salas de bate-papo e os sistemas *wiki*, usados para gerenciamento colaborativo de informações.

2.1.2.6 Sistemas sociais

Em princípio a *internet* tinha como característica o anonimato dos usuários, mas havia uma tendência a se ter uma rede onde pessoas se conheçam. Então surgiram os sistemas sociais, onde pessoas registram informações pessoais, conhecem pessoas e podem fazer buscas de indivíduos com perfil desejado. Entre esses sistemas, estão os *weblogs*, para registro e gerenciamento de informações pessoais e páginas de cadastro pessoal compartilhado.

2.1.3 Características de aplicações *web*

Aplicações *web* têm peculiaridades e características próprias que vão além do fato de usarem a *internet* como plataforma de trabalho. Muitas delas são comuns a aplicações tradicionais, outras são adaptadas e ainda outras são exclusivas. Características comuns englobam o escopo do desenvolvimento, usabilidade e produto, e características próprias das aplicações *web* fazem parte do escopo da evolução, como pode ser visto na figura 2.1. Todos esses aspectos, citando características relevantes de *softwares* tradicionais, serão abordados a seguir.

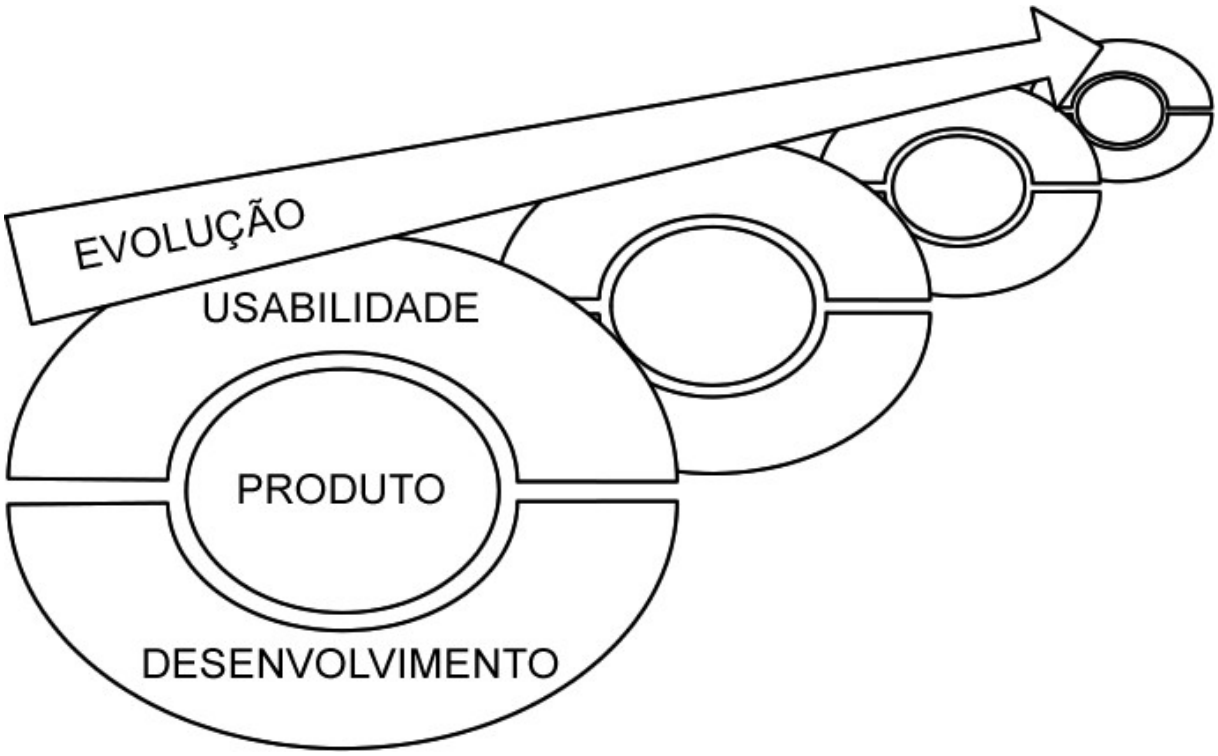

Figura 2.1 – Dimensões das aplicações *web*

2.1.3.1 Desenvolvimento

Muitas características no desenvolvimento de aplicações *web* que as diferenciam de aplicações convencionais devem ser observadas, como o grupo de desenvolvedores, infra-estrutura, processo e integração.

Na questão do grupo de desenvolvimento, observa-se a diversidade e a ênfase do conhecimento entre integrantes. Aplicações *web* envolvem programação, conhecimento de redes, desenho artístico, relações públicas, entre outros. Por isso existe a importância da

multidiciplinaridade entre pessoas do grupo e um conhecimento geral dos coordenadores de projetos de aplicações *web*. Outra prática característica na criação é o desenvolvimento em comunidade, onde pessoas do mundo inteiro participam do processo de autoria de uma aplicação.

A infra-estrutura técnica no meio de aplicações *web* é caracterizada pelo fato de as tecnologias envolvidas serem heterogêneas e recentes. Aplicações *web* dependem de dois fatores básicos: o cliente e o servidor. Devido à diversidade de plataformas de navegação, as aplicações nem sempre executam da mesma forma para os clientes, em contraste a *softwares* dependentes de apenas um computador ou rede local. Devido à pressão do mercado, as tecnologias usadas tendem a ser imaturas, pois são desenvolvidas às pressas e muitas vezes abandonadas antes de serem terminadas.

O processo de concepção de aplicações *web* exige extrema flexibilidade e um paralelismo bem coordenado. A flexibilidade é necessária, pois além de modificações solicitadas pelo dono da aplicação, há uma massiva quantidade de sugestões e críticas dos usuários devido à interatividade das aplicações *web* e também por causa da constante atualização e criação de tecnologias para *internet*. Por esses mesmos fatores citados, é necessário o trabalho paralelo coordenado dos desenvolvedores, para atender aos requisitos solicitados em um período aceitável pelos usuários e clientes.

Um ponto forte que faz o desenvolvimento de aplicações *web* especial é a integração, tanto interna quanto externa. A integração interna é a capacidade de um sistema *web* assimilar e interagir com sistemas e tecnologias legadas, como bancos de dados antigos e catálogos de mídia impressa. A integração externa trata da ligação com sistemas do mundo inteiro. Essa integração é importante por que sistemas *web* distintos com esquemas e modelos de dados distintos muitas vezes precisam interagir, e como sistemas *web* têm conteúdo dependente de outras fontes, uma ou mais fontes podem estar perdidas ou modificadas.

2.1.3.2 Usabilidade

Comparado com aplicações tradicionais, a usabilidade de aplicações *web* é extremamente heterogênea (KAPPEL, 2003). A cultura, grau de instrução, religião, gostos, acessibilidade, capacidades físicas, entre outros aspectos são muito variados, assim como mudanças nesses aspectos são constantes. Além de tudo, a infra-estrutura da rede pode ser diferente em vários lugares, não podendo assim se garantir o acesso em momentos

determinados. Esses fatores contextuais são tão delicados que foram divididos em três categorias: contexto social, técnico e natural.

Uma característica marcante na internet é a espontaneidade do usuário, uma vez que ele pode entrar, sair e ir para outra página, a hora que desejar. Por isso, muitos fatores devem ser observados, como por exemplo, o tempo de *download* e o impacto visual da página principal. Ainda no contexto social, a diversidade cultural é um fator extremamente relevante na criação de uma aplicação *web*, pois existem usuários com diferentes tipos de habilidades, preferências e conhecimentos.

O contexto técnico é extremamente variável, tanto na infra-estrutura física do ponto de acesso do usuário quanto nos *softwares* usados para utilização da aplicação. Apesar da tecnologia de comunicação e de redes ter evoluído a ponto de se ter grande velocidade de transferência de dados, por questões sociais e econômicas, existem muitos usuários que possuem baixa largura de banda, e por essa razão, esse fator deve ser levado em consideração para garantir a qualidade de serviço. Assim como existem diversos *softwares* diferentes que realizam funções similares, o mesmo se aplica aos navegadores. Porém, ainda não existe um acordo entre desenvolvedores sobre um padrão para interpretação de dados da *internet*, ocasionando assim muitos problemas de compatibilidade com navegadores. Por esse motivo, desenvolvedores tendem a escrever suas aplicações usando o máximo possível de elementos comuns entre os navegadores mais usados.

O fato de a *internet* ser uma rede que liga o mundo inteiro implica que as aplicações *web* devem considerar fortemente fatores como a globalidade e disponibilidade. Muitas línguas são faladas no mundo, por isso, aplicações devem ter versões em diversos idiomas no caso de seu escopo ser globalmente distribuído. E como qualquer pessoa pode ter acesso à *internet*, é importante lidar com aspectos de segurança, pois intencionalmente ou acidentalmente, o sistema pode ser acessado e modificado indevidamente. Outro fator importante é a disponibilidade, que se refere ao tempo em que uma aplicação pode ser acessada. Aplicações com disponibilidade permanente devem poder ser acessadas a qualquer dia e a qualquer hora.

2.1.3.3 Produto

A aplicação *web* em si é dividida em três componentes básicos: o conteúdo, o hipertexto e a apresentação. Essa divisão é extremamente importante, pois ajuda a organizar o projeto da aplicação através dessa separação de elementos.

Uma aplicação *web* deve ter conteúdos em formatos e qualidades aceitáveis. Dependendo do propósito da aplicação, o conteúdo pode consistir em textos, imagens, vídeos e sons. Porém, muitas aplicações têm intenso número de acessos assim como atualizações, como por exemplo, *sites* de notícias. O grande desafio é poder garantir a qualidade dos dados apesar do grande volume e alta freqüência de atualizações (KAPPEL, 2003).

A navegação em uma aplicação *web* é feita através do hipertexto. Hipertextos têm como elementos básicos: nós, que são unidades de informação; *links*, ou caminhos que vinculam um nó a outro; e âncoras, que são regiões (texto ou imagem) de um nó que referencia ou serve de destino para um *link*. Uma importante característica do hipertexto é a não-linearidade, o que implica que a navegação entre os nós não são necessariamente seqüenciais, possibilitando o acesso a informações através de associações de imagens e palavras-chave. Outras características que merecem especial atenção no projeto do hipertexto são a desorientação, onde o usuário perde o entendimento contextual da página, e a sobre-carga cognitiva, que ocorre quando o usuário precisa memorizar grandes seqüências de acessos a nós. Uma forma de manter uma boa navegabilidade é usar textos e imagens bem associadas a destinos em *links*.

Um elemento crucial nas aplicações em geral é a interface de usuário ou apresentação, pois tem tanto em comum com o estudo de pessoas como com questões tecnológicas (PRESSMAN, 1995). Porém, a aplicação desse elemento é ainda mais delicada em aplicações *web*, levando em consideração a estética e a facilidade de entendimento. A aparência de uma página e a impressão que ela passa para o usuário pode ser a diferença entre sites bem sucedidos e os que fracassam. Por isso a página deve ter uma aparência simples, mas que passe a idéia do sistema e seja atraente aos olhos. A facilidade de uso ou auto-explicação também são fatores extremamente importantes, pois uma aplicação difícil de usar e entender pode ser imediatamente encerrada pelo usuário. A interface deve ser o mais simples possível, permitindo que o usuário rapidamente torne-se familiarizado ao uso.

2.1.3.4 Evolução

Todos os aspectos de aplicações *web* citados sofrem constantes mudanças. Os sistemas estão sempre em desenvolvimento devido à contínua mudança de condições; a interface é adaptada por mudanças de contextos sociais; e o produto da aplicação deve ser modificado de acordo com o contexto tecnológico. Essas condições definem a característica única das aplicações *web*: a evolução. A evolução rege todos os outros aspectos mencionados e é caracterizada pela mudança contínua, devido à variação de condições da infra-estrutura tecnológica e de requisitos e pela pressão do mercado, que exige dos desenvolvedores agilidade

na modificação ou criação de sistemas. Enquanto que para *softwares* convencionais evolução signifique uma série de versões, em aplicações *web* a evolução é contínua (KAPPEL, 2003).

2.2 Engenharia *web*

2.2.1 Introdução

No início da era da *internet*, aplicações *web* se limitavam a páginas informativas, orientadas a documento. Hoje, aplicações *web* são complexos sistemas de *software* que provêm serviços interativos, customizáveis e com grandes quantidades de dados, acessíveis por diversos dispositivos diferentes. Porém, freqüentemente o desenvolvimento de aplicações *web* é encarado como um evento de um só passo e espontâneo, e comumente baseado nos conhecimentos, experiências e práticas que diferem a cada indivíduo, limitando o reuso ao paradigma do "copiar e colar", e caracterizada pela má documentação das decisões de projeto (KAPPEL, 2003).

Os fatores que motivam a criação da engenharia *web* são:

- Abordagem orientada ao documento: apesar das páginas hoje em dia não serem essencialmente informativas, o processo de desenvolvimento ainda é voltado à criação de documentos e de vínculos entre eles. Apesar de que a autoria tem essencial papel na criação de sistemas *web*, uma visão orientada apenas à criação de informação não é adequada;
- Supõe-se a simplicidade no desenvolvimento: os desenvolvedores *web* consideram fácil a criação de aplicações *web* e esse fato é reforçado pela presença de aplicativos de edição e geração para páginas de *internet*, resultando em redundância e inconsistência;
- Conhecimentos da engenharia de *software* tradicional nem sempre podem ser aplicados: apesar de características comuns entre *softwares* convencionais e aplicações *web*, existem elementos únicos constantes em aplicações *web* já citados, nos quais a aplicação da engenharia de *software* não é adequada.

Em conseqüência dessa visão, o resultado é a criação de aplicações *web* com carência de funcionalidades, desempenho e qualidade. Além disso, as aplicações se tornam complexas, difíceis e caras de gerenciar e atualizar.

O processo de desenvolvimento de uma aplicação *web* passa por fases de ciclo de vida similares aos da engenharia de *software* convencional. Apesar disso, é importante tratar a engenharia *web* como uma disciplina separada, pois se definindo um campo de estudo,

pesquisas e desenvolvimentos na área, aplicações para *internet* se tornam mais promissoras e produtivas, considerando sua importância e carências de métodos para desenvolvimento *web*.

De acordo com Selmi (Selmi, 2005) os princípios básicos da engenharia *web* são similares aos da engenharia de *software*:

- Definir claramente objetivos e requerimentos;
- Desenvolvimento sistemático em fases de uma aplicação *web*;
- Planejar cuidadosamente essas fases;
- Gerenciar continuamente o processo de desenvolvimento inteiro.

De acordo com o exposto e com (Deshpande, 2001) podemos definir engenharia *web* como:

1) Engenharia *web* é a aplicação de abordagens sistemáticas e quantificáveis para eficiente análise de requisitos, projeto, implementação, teste, operação e manutenção de aplicações *web* de alta qualidade.
2) Engenharia *web* também é a disciplina científica voltada ao estudo dessas abordagens.

Existem muitos métodos de engenharia *web* usados e em constante processo de aprimoramento. A figura 2.2 mostra a criação desses métodos no decorrer dos últimos anos e os métodos consagrados nos quais foram baseados.

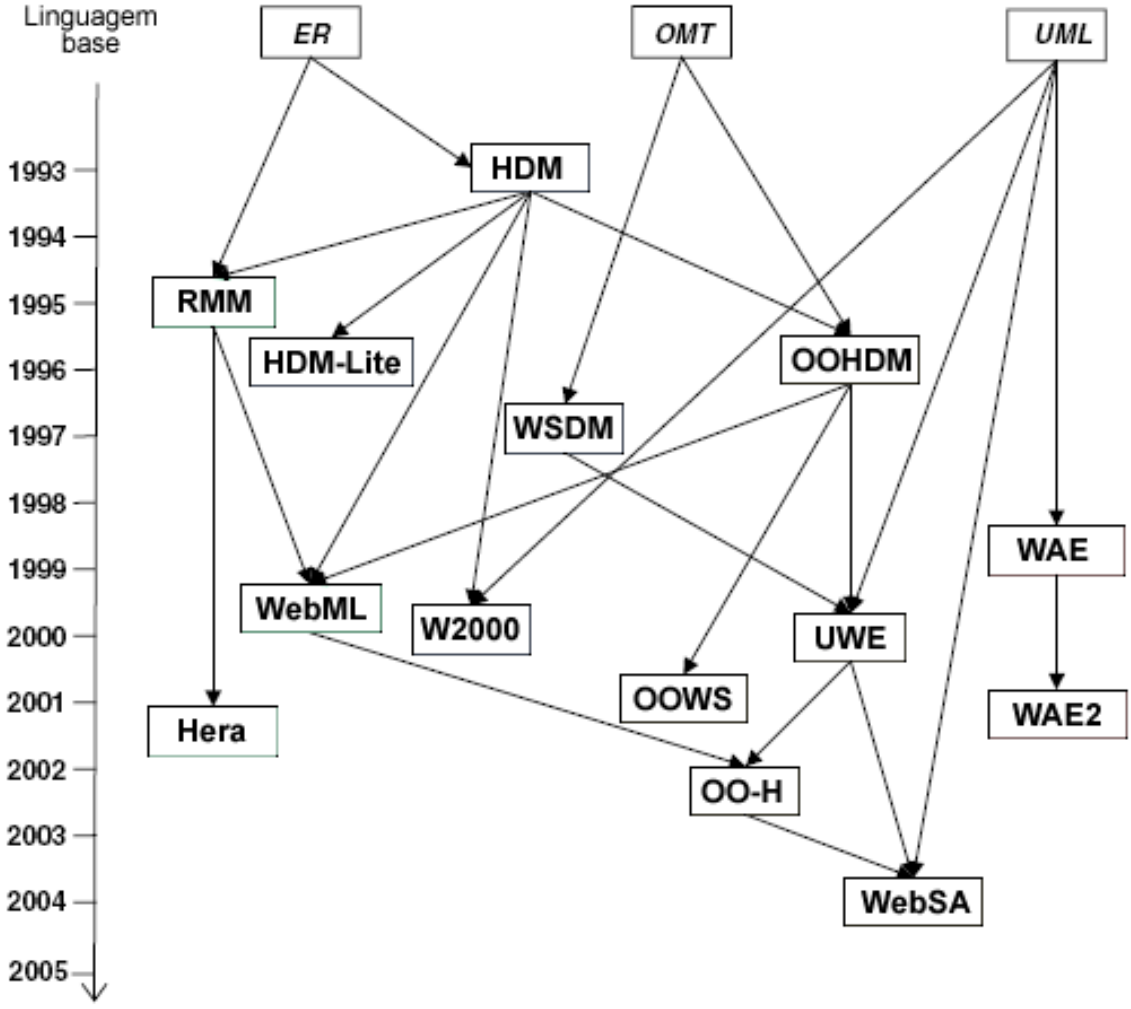

Figura 2.2 – Métodos de desenvolvimento *web*

2.2.2 O processo de desenvolvimento *web*

Entre as características das aplicações *web* que tornam seu desenvolvimento difícil e unicamente desafiante incluem interação em tempo real, complexidade, flexibilidade e o desejo de prover informação personalizada. Além disso, o tempo e o esforço para projetar e desenvolver uma aplicação *web* é difícil de estimar com precisão razoável (MURUGESAN, 2005).

Através de estudos e de experiências próprias, pesquisadores e desenvolvedores tendem a usar como meio de desenvolvimento de aplicações *web* o processo evolucionário, pelas seguintes razões:

- Auxilia desenvolvedores a entenderem o contexto no qual a aplicação será empregada e usada;
- Facilita a comunicação entre pessoas envolvidas no processo;
- Dá suporte a contínua evolução e manutenção;
- Facilita o gerenciamento de conteúdo;
- Ajuda a gerenciar corretamente a complexidade e diversidade do processo de desenvolvimento.

A figura 2.2 mostra o diagrama dos elementos que compõem o processo evolucionário e a relação entre eles. Nas subseções a seguir serão descritos esses elementos.

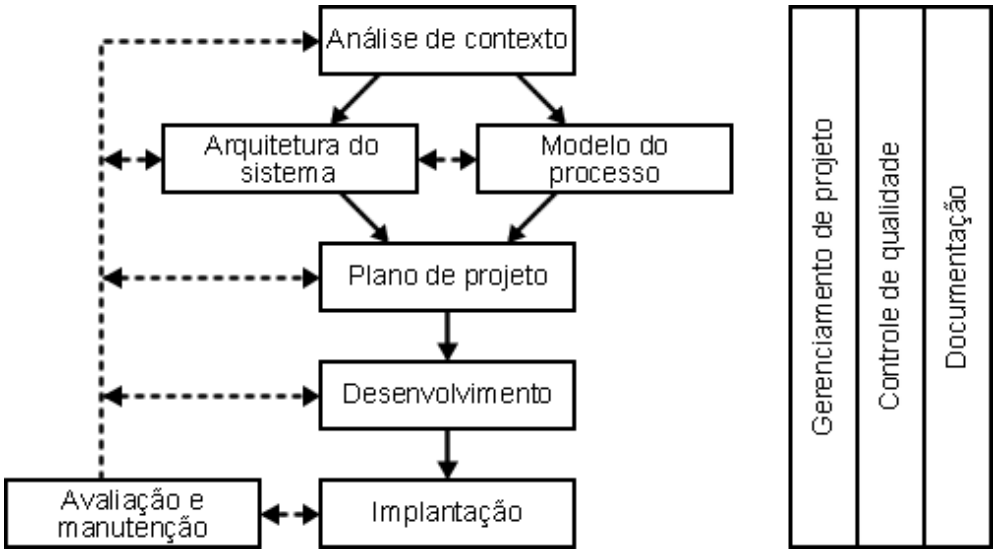

Figura 2.3 – Diagrama do processo evolucionário de desenvolvimento *web*

2.2.2.1 Análise de contexto

A análise de contexto é a primeira etapa do processo de desenvolvimento, crucial para o sucesso da aplicação final. Essa etapa engloba atividades tais como a análise, onde os objetivos e requisitos são pesquisados; a documentação, que compõe os registros da análise; a verificação, onde é verificado se os requisitos foram descritos corretamente; a validação, onde é visto se todos os requisitos relevantes foram listados; e o gerenciamento de requisitos, que engloba as outras atividades citadas, modificando-as conforme mudanças ou acréscimos de requisitos. Uma nota sobre a importância da análise de contexto é que uma falha detectada fora da análise de contexto custa em torno de 200 vezes mais para ser tratada (Boehm, 1981).

Na engenharia *web*, a análise de contexto segue princípios que tratam de requisitos funcionais, de conteúdo, de qualidade, do ambiente de sistema, de interface de usuário e restrições de projeto.

Os requisitos funcionais tratam das funções e capacidades da aplicação. Para uma boa análise funcional, é necessário um estudo sobre as necessidades do ambiente onde será implantada a aplicação e, no caso de haver um sistema legado, verificar suas funcionalidades e restrições. Através de protótipos é possível testar as funcionalidades e verificar se as mesmas correspondem à expectativa dos usuários e clientes.

A análise do conteúdo da futura aplicação é importante, pois através dela será definido o tipo e o contexto do conteúdo. Nessa fase é feito o estudo dos formatos de arquivos do sistema legado, podendo esse ser uma outra aplicação, convencional ou *web*, ou ainda baseado em métodos não informatizados, como arquivos de fichários.

Além dos requisitos funcionais, também existem os requisitos não-funcionais, que tratam da qualidade geral da aplicação. Com relação as suas capacidades, um sistema *web* de qualidade deve ser seguro e ter funcionalidades corretas. Um sistema também dever ser confiável, ou seja, estar acessível e ter um bom grau de recuperação em caso de falhas. A usabilidade, eficiência, e portabilidade também caracterizam a qualidade, sendo um importante aspecto a ser considerado na análise de contexto.

As notações usadas em documentos de análise de contexto que podem ser usadas na engenharia *web* são histórias, itens de requisitos, descrição formatada e especificação formal. Histórias são documentos com uma descrição informal sobre os requisitos e objetivos de usuários e clientes. São pouco trabalhosas para elaborar, não exigem formalidades e têm uma razoável flexibilidade, porém são difíceis de serem verificadas e são pouco precisas. Itens de requisitos é uma forma mais organizada de descrever o contexto, consistindo em uma lista de requisitos numerados. Sua precisão é precária, mas sua verificação é um pouco mais fácil. Documentos com descrição formatada seguem padrões para descrição de requisitos através de uma organização dos atributos e características de cada requisito. Possuem boa precisão, são flexíveis, fáceis de validar, e exigem um esforço razoável para composição e formalização. Por fim, outra forma de registrar o contexto é a especificação formal, que é escrita através de linguagens formais com sintaxe e semântica bem definidas. São pouco flexíveis, exigem conhecimento técnico e muito esforço, porém são precisas e facilmente validadas.

A etapa da análise de contexto definirá os elementos que farão parte da arquitetura do sistema.

2.2.2.2 Projeto da arquitetura do sistema

A arquitetura de um sistema é o conjunto dos elementos que o compõe e a relação entre eles. A etapa do projeto da arquitetura do sistema pode ser vista como uma transição entre a análise de contexto e a implementação do sistema, e é responsável pela definição dos elementos da arquitetura com base no levantamento dos requisitos feitos na etapa de análise.

A decisão sobre os elementos que farão parte da arquitetura do sistema deve ser tomada com base em três aspectos relativos a aplicações *web*, que são a estrutura física usada, como os servidores, as características da própria aplicação, com seus elementos e funcionalidades, e finalmente os softwares envolvidos para execução, manutenção e outras funções para o desenvolvimento e implantação da aplicação.

O esforço aplicado no projeto da arquitetura do sistema varia com o tipo de aplicação a ser desenvolvida. No caso de projetos de sistemas a serem implantados com tecnologia legada, será levado em consideração os elementos da arquitetura antiga, que deve ser adaptada aos elementos arquiteturais da nova aplicação para compor um esquema de arquitetura.

Além de serem verificados meios para cumprir os requisitos gerados na análise de contexto, durante a escolha dos elementos arquiteturais deve ser levada em consideração a decisão sobre o modelo do processo, pois a escolha do modelo é fortemente dependente da arquitetura do sistema.

2.2.2.3 Modelo do processo

Durante a definição da arquitetura do sistema, deve-se considerar a escolha do melhor processo a ser aplicado para a concepção da aplicação *web*. O modelo do processo é responsável pela organização do desenvolvimento do sistema, e define normas e padrões para as fases de desenvolvimento, para dessa forma facilitar a compreensão da estrutura geral da aplicação, assim como a comunicação entre os desenvolvedores.

O modelo de um processo também define métodos para realizar cada fase de desenvolvimento assim como a transição entre essas fases. Um método definido num modelo de processo trabalha com questões locais das fases do projeto, enquanto o modelo de processo aborda questões globais em todo o desenvolvimento da aplicação. A figura 2.4 mostra um gráfico que representa o modelo do processo de desenvolvimento de aplicações *web*.

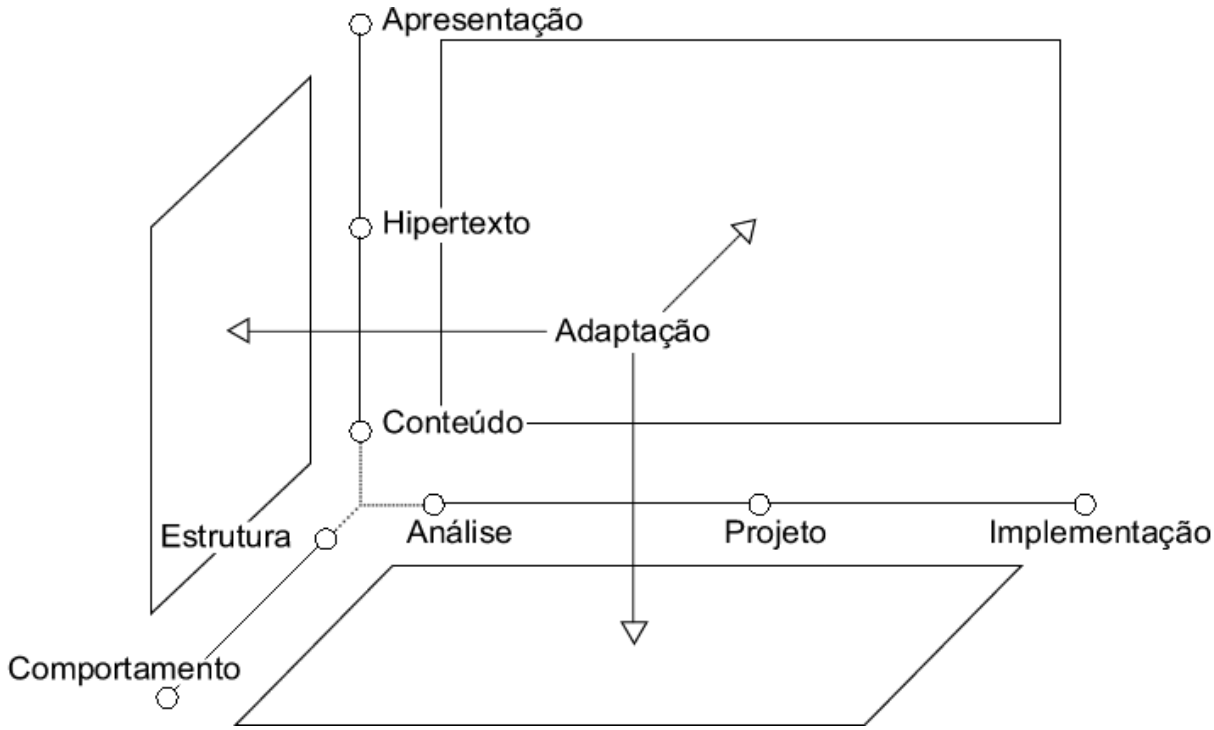

Figura 2.4 – Modelo do processo de desenvolvimento *web*

2.2.2.4 Plano de projeto

Com a arquitetura do sistema bem definida, e feita a escolha do modelo de processo a ser aplicado, vem a etapa de planejamento do projeto. O plano de projeto consiste nos tipos de atividades e quando as mesmas serão realizadas. O objetivo é o melhor aproveitamento possível de recursos e tempo, de forma que a aplicação seja concebida dentro do prazo e com funcionalidades satisfatórias. O plano de um projeto deve levar em consideração a constante mudança das tecnologias envolvidas assim como os requisitos do cliente, o número e o comportamento dos usuários.

Outro fator importante que caracteriza um projeto bem planejado é o paralelismo de atividades de forma coordenada para que as atividades multidisciplinares resultem em módulos de sistema que levariam mais tempo para serem feitos se seus elementos fossem trabalhados sequëncialmente.

2.2.2.5 Desenvolvimento

A partir do plano do projeto, inicia-se a fase do desenvolvimento da aplicação *web*. Nessa fase são implementados os módulos que compõe a aplicação através da programação e são definidos os elementos navegacionais, o conteúdo da aplicação e o aspecto visual das páginas. As linguagens de programação usadas são de dois tipos, considerando o escopo de execução dos módulos, que são linguagens para programas do lado do servidor (*server-side*) ou para programas do lado do cliente (*client-side*).

Os módulos do lado do servidor são responsáveis pela maior parte funcional da aplicação, que consiste em gerenciamento de banco de dados, servidor *web* e servidores de arquivos. Muitos aspectos inerentes à segurança devem receber atenção especial no lado do servidor, pois se trata da interface comum a todos os usuários do sistema, sendo que dessa forma qualquer problema de nível de servidor pode ser visível a todos os usuários.

Os módulos do lado do usuário são executados na máquina do usuário e são responsáveis por funcionalidades locais que, quando bem aplicadas, podem facilitar a utilização da aplicação *web* e também abstrair ou serem combinadas com funcionalidades do próprio lado do servidor, aumentando a segurança e satisfação do usuário. O desenvolvimento básico *client-side* não é feito com o mesmo esforço e codificação do desenvolvimento *server-side* porém, a longo prazo, pode exigir muito mais esforço e tempo por se tratar de módulos que executam em um ambiente muito mais heterogêneo que o servidor. Além de mudanças que caracterizam o número e o comportamento dos usuários, há também uma variação constante de *softwares* navegadores, que mudam constantemente em um mesmo computador e tendem a ser diferentes entre computadores, forçando a equipe de desenvolvimento a lidar com essas constantes mudanças através de manutenção, e tratam as diferenças entre navegadores dos usuários aplicando código extra e códigos comuns ao maior número de navegadores e sistemas operacionais possível.

Apesar das diferenças de especializações entre os desenvolvedores de uma equipe, uma coordenação de desenvolvimento é crucial, para que haja harmonia entre programas de cliente, programas de servidor e interface de usuário.

2.2.2.6 Implantação

A fase de implantação é a etapa em que a aplicação *web* é posta à disposição dos usuários *on-line*. A implantação consiste na instalação da aplicação no servidor e no uso efetivo pelos usuários. Quando a aplicação for implantada, deve-se considerar fatores como compatibilidade de estrutura tecnológica, qualidade do servidor e escopo da aplicação.

O fator da compatibilidade se refere à capacidade do servidor no qual a aplicação será implantada para preencher os requisitos impostos pela mesma. Entre os requisitos básicos estão a largura de banda, pois ela determinará a velocidade de acesso pelos usuários, e deverá ser escolhida de acordo com o tipo de aplicação. Aplicações com intenso fluxo de dados, como aplicações colaborativas e baseadas em fluxo de trabalho, devem ter como prioridade durante a implantação uma razoável largura de banda. Já páginas informativas e interativas com acessos moderados não têm a largura de banda como um fator crucial.

A qualidade do servidor é um critério muito importante no momento da implantação, pois devem ser avaliadas as capacidades do servidor com relação às funcionalidades da aplicação, como versões de linguagens dinâmicas suportadas, limite para o armazenamento de dados e serviços de *e-mail*.

Os escopos de aplicação relevantes para a implantação de um sistema *web* consistem no grupo de usuários que irá utilizar a aplicação, que pode se tratar de um escopo local ou global. Escopo local de implantação significa que um grupo restrito de usuários irá trabalhar com a aplicação, assim tornando mais fácil o tratamento com a compatibilidade tecnológica, porém, geralmente implantações locais são feitas em ambientes de trabalho, tendo assim que ser feito um treinamento de pessoal para o uso da aplicação. O escopo global se refere à implantação de sistemas disponíveis na *internet* para usuários do mundo inteiro, sendo que um controle global é necessário para estimar a efetiva implantação e uso do sistema.

2.2.2.7 Avaliação e manutenção

Depois de a aplicação ser empregada para o uso *on-line*, é necessária a avaliação e manutenção do sistema para que o mesmo se mantenha disponível e funcional, assim como para manter o sistema atualizado de acordo com as mudanças tecnológicas e demanda dos usuários. Para isso é necessário formular políticas e procedimentos de manutenção de conteúdo, baseados nas decisões tomadas durante a etapa do projeto da arquitetura do sistema sobre como o conteúdo informativo deveria ser gerenciado, e então implementá-los.

É importante periodicamente revisar sistemas e aplicações baseados na *web* devido à constante circulação de conteúdo, riscos de segurança em potencial, desempenho do sistema e padrões de uso, e assim tomar medidas corretas para atualizações.

A manutenção e constante avaliação dos sistemas *web* merecem uma concentração adicional com relação ao desenvolvimento de *softwares* convencionais, visto que a demanda por tratamento de mudanças é muito mais freqüente em um ambiente *on-line*.

2.2.2.8 Gerenciamento de Projeto

O objetivo principal do gerenciamento do projeto é garantir que os processos e atividades trabalhem em harmonia. A criação de aplicações *web* de qualidade requer uma boa coordenação entre os diversos esforços envolvidos no ciclo de desenvolvimento. Porém, muitos estudos revelam que um mau gerenciamento de projeto é a maior causa de falhas durante o desenvolvimento, que por sua vez se reflete em um produto final de má qualidade. Um gerenciamento de projeto ruim prejudica uma boa engenharia, por isso um bom gerenciamento de projeto pode definir o sucesso de uma aplicação *web*.

Gerenciar um ambiente de desenvolvimento *web* complexo e extenso é uma tarefa desafiadora, que requer do gerente técnicas multidisciplinares e, muitas vezes, técnicas diferentes daquelas usadas no desenvolvimento de *software* tradicional.

Como o gerenciamento de projeto trata da coordenação de todo o processo de desenvolvimento, ele está presente em todas as outras fases do desenvolvimento evolucionário.

2.2.2.9 Controle de qualidade

Mesmo que uma fase do processo de desenvolvimento evolucionário não resulte necessariamente em um produto final, ainda assim o resultado de cada fase deve preencher requisitos mínimos para prosseguir para outra fase do processo, para dessa forma obter um desenvolvimento coordenado e de qualidade. Para fazer a avaliação de qualidade da componente resultante de cada fase do processo existe uma atividade paralela a todas as fases, que é o controle de qualidade.

Se não houvesse um constante controle de qualidade das etapas do processo, o número de iterações dentro do ciclo de desenvolvimento seria muito maior, visto que a qualidade seria avaliada apenas na fase de avaliação, perdendo assim a agilidade da obtenção do produto final.

2.2.2.10 Documentação

Durante o processo de desenvolvimento *web*, é interessante fazer um registro sobre requerimentos, resultados e codificação, para facilitar a avaliação dos desenvolvedores para manutenção, para facilitar o uso de aplicações complexas, registrar e classificar requisitos. A atividade de documentação é responsável por essa tarefa. Através da documentação é possível organizar as atividades do processo com o uso de textos e diagramas, de modo que seja de fácil compreensão para os envolvidos no projeto e para usuários de sistemas complexos, cuja auto-explicação é inviável.

3 ENGENHARIA WEB BASEADA NA UML

Nos capítulos anteriores foram evidenciadas características relativas às aplicações *web* que as diferenciam de outros *softwares*, tanto quantitativa como qualitativamente. Também foi

abordado o tema da engenharia de software, especificamente para projetos de aplicações *web*, enfatizando suas peculiaridades e diferenças com relação a métodos tradicionais. Foram vistos também os elementos básicos que compõem o processo de desenvolvimento *web*, assim como as características de suas atividades.

Nesse capítulo, será introduzido um processo de desenvolvimento de aplicações *web* que é explorado e trabalhado atualmente por muitos pesquisadores e utilizado com boa aceitação entre desenvolvedores *web*, que buscam uma maneira formal e organizada de criar sistemas de *internet*: a engenharia *web* baseada na UML, conhecida também como UWE.

3.1 Introdução à UWE

UWE é um acrônimo em inglês para Engenharia *Web* baseada na UML. A UWE trata-se de uma abordagem para o desenvolvimento sistemático de aplicações *web*, e tem como principal objetivo a unificação da abstração de alto nível do processo de desenvolvimento. As principais razões para a unificação são:

- Melhor integração e comparação de elementos do modelo;
- A codificação semi-automática a partir do modelo;
- Permitir diferentes processos de desenvolvimento;
- Definição de subconjuntos de elementos de modelagem;
- Permitir uso de notações diferentes.

De acordo com os seus objetivos, a UWE é baseada na UML, linguagem para descrição de construções e relacionamentos de sistemas complexos (DOUGLASS, 1998), e no Processo Unificado UP, e é compatível com a arquitetura de modelagem OMG.

3.1.1 O processo UWE

As principais características do processo de desenvolvimento de aplicações *web* baseadas na UWE são a divisão do problema nas etapas iniciais e a implementação do processo de desenvolvimento orientado ao modelo.

A UWE trabalha com uma abordagem em que os diversos aspectos que descrevem sistema *web* são trabalhados separadamente. As etapas do processo de desenvolvimento, como a análise de requisitos, definição da arquitetura e projeto são descritas através de módulos para

diferentes pontos de vista do sistema, que são o conteúdo, a estrutura do hipertexto e a interface com usuário. Em cada etapa do processo e a faceta abordada, é dividida também a descrição comportamental da descrição estrutural de cada módulo. Tudo isso facilita a divisão de tarefas da equipe de desenvolvimento e permite que conhecimentos específicos sejam bem aplicados a cada parte do problema geral.

O desenvolvimento orientado a modelos permite que a visão geral dos componentes da aplicação e o estudo para manutenção ou mudanças do sistema sejam facilitados. Além disso, a orientação ao modelo enfatiza a transição entre as etapas do processo de desenvolvimento. O uso correto dos elementos é imprescindível para a geração automática de código com base no modelo.

3.1.2 A notação UWE

A UWE usa modelos visuais que são usados para criar a documentação e facilitar o processo de desenvolvimento. O uso de notação visual permite a construção de modelos bem expressivos e ao mesmo tempo intuitivos, empregando uma semântica formal e facilitando o aprendizado dos desenvolvedores.

Os estereótipos usados para representar elementos dentro dos modelos da UWE são representados através de texto limitado por ‘<<’ e ‘>>’ ou ícones com ilustrações intuitivas para a função do estereótipo. A tabela 3.1 relaciona os estereótipos usados nos modelos UWE, suas classificações na UML, os módulos em que são usados e os ícones que os representam.

Estereótipo UWE	Classe UML	Classe UML	Ícone
<<anchor>>	classe	modelo de apresentação	
<<anchored collection>>	classe	modelo de apresentação	
<<button>>	classe	modelo de apresentação	
<<choice>>	classe	modelo de apresentação	
<<form>>	classe	modelo de apresentação	
<<guided tour>>	classe	modelo de navegação	
<<image>>	classe	modelo de apresentação	
<<index>>	classe	modelo de navegação	
<<menu>>	classe	modelo de navegação	
<<navigation class>>	classe	modelo de navegação	
<<navigation link>>	associação	modelo de navegação	
<<navigation property>>	propriedade	modelo de navegação	
<<page>>	classe	modelo de apresentação	
<<presentation class>>	classe	modelo de apresentação	
<<presentation group>>	classe	modelo de apresentação	
<<presentation property>>	propriedade	modelo de apresentação	
<<process class>>	classe	modelo de navegação/proceso	
<<process link>>	associação	modelo de navegação	
<<process property>>	propriedade	modelo de navegação/proceso	
<<query>>	classe	modelo de navegação	
<<text imput>>	classe	modelo de apresentação	
<<textt>>	classe	modelo de apresentação	
<<user action>>	associação	modelo de processo	

Tabela 3.1 – Estereótipos e notação simbólica da UWE

3.2 O meta-modelo UWE

Um meta-modelo tem como função a descrição dos elementos de um modelo e a relação entre eles. O meta-modelo UWE é definido através da integração do meta-modelo UML com elementos específicos do modelo UWE.

O meta-modelo UWE consiste na adição de dois níveis principais ao meta-modelo UML, que são o módulo de núcleo e o módulo de adaptabilidade. O módulo de núcleo é composto pelo módulo de requisitos, de conteúdo, de navegação, de apresentação e de processo. Esses módulos são estruturados em uma relação de dependência e refletem a divisão do problema em partes. O módulo de adaptabilidade é dependente do módulo de núcleo e representa os aspectos de adaptação em manutenção do processo. A figura 3.1 apresenta a estrutura do meta-modelo UWE e a relação entre os módulos.

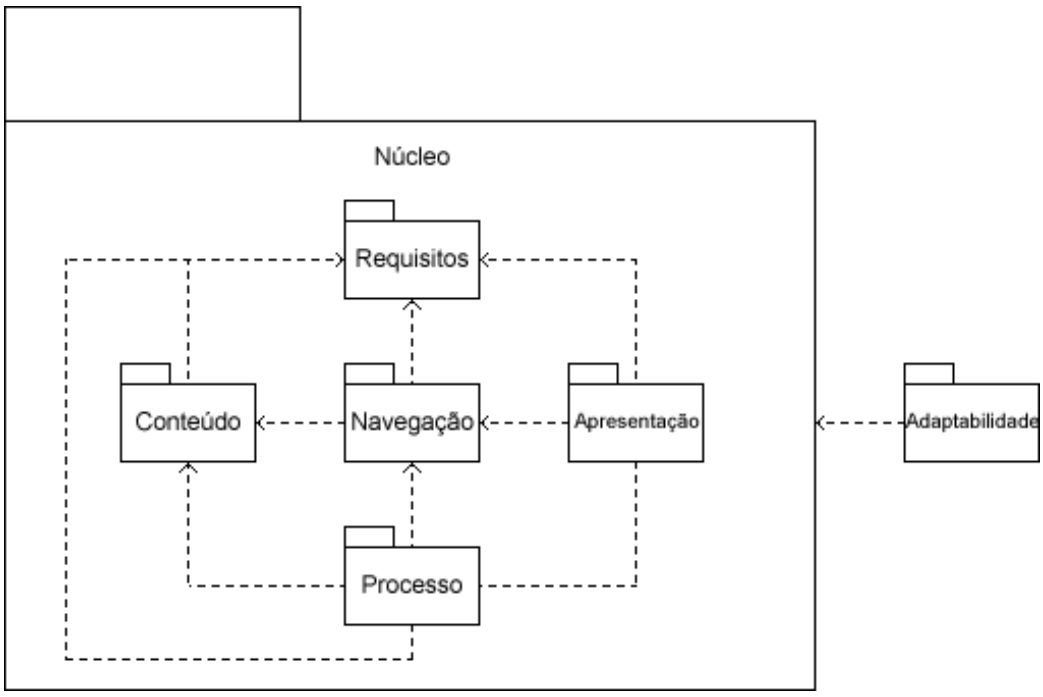

Figura 3.1 – Diagrama estrutural do meta-modelo UWE

A seguir serão descritas as características dos elementos do meta-modelo UWE.

3.2.1 Módulo de requisitos

O módulo de requisitos consiste na extensão dos modelos de casos de uso da UML, através da distinção entre os processos de negócios e processos de navegação e a personalização dos casos de uso e extensões para os diagramas de atividade. Assim como na análise de requisitos modelados por UML, a UWE também usa descrições de casos de uso, para descrever a forma que o sistema se comporta para satisfazer um requisito (HAMILTON, 2006).

3.2.2 Módulo de conteúdo

A modelagem de conteúdo em processos de desenvolvimento de aplicações *web* não apresenta diferenças com a modelagem de conteúdo de outros *softwares*. Então são usados os mesmos elementos do modelo UML para criar descrição estrutural do diagrama de classes, associações e módulos. A modelagem comportamental do conteúdo pode ser feita usando elementos UML como máquina de estados e diagramas de seqüência.

3.2.3 Módulo de navegação

O módulo de navegação da UWE é composto basicamente por nós e vínculos representados por diagramas de classes e a relação entre as classes. Um conjunto de subclasses derivadas das classes nó e vínculo completam o modelo de navegação através de meta-classes, como a classe de navegação e a classe de processo e seus respectivos vínculos de navegação e vínculos de processo, classe de menu, primitivas de acesso para índices, roteiro guiado e consultas.

A meta-classe nó pode ser encarada como um nó num grafo de navegação, o que significa que quando um nó é acessado durante a navegação, o usuário recebe informações e a possibilidade de tomar novas decisões. Porém, um nó não é necessariamente uma página da aplicação *web*. O módulo de apresentação é responsável pela descrição das páginas.

Os atributos de um objeto da classe nó são:

- *isLandmark*, que armazena verdadeiro ou falso, indicando se o nó é acessível a partir de qualquer outro nó do grafo de navegação;
- *isHome*, que também armazena um valor booleano, indicando se o nó é o nó inicial do grafo de navegação.

As associações possíveis na classe nó são:

- *inLinks*, que é uma lista de vínculos do modelo de navegação que conduzem ao nó;
- *outLinks*, que consiste em uma lista de nós acessíveis a partir do nó em questão.

Vínculos são arestas que ligam dois nós de um grafo de navegação, com sua generalização representada através de associações em UML. Os vínculos não representam necessariamente a transição entre páginas causada pela ação do usuário. As representações de vínculos postos à disposição de ações do usuário fazem parte do módulo de apresentação.

Os atributos de um vínculo são:

- *isAutomatic*, que especifica se a ação de transição entre nós depende ou não da ação do usuário;

As associações possíveis em vínculos são:

- *source*, atributo que indica nó onde se origina o vínculo;
- *target*, que referencia o nó de destino do vínculo.

A classe de navegação representa os nós navegáveis com estrutura de hipertexto e estabelece a conexão entre o modelo de navegação e o modelo de conteúdo. Um objeto da classe de navegação que é associado a um objeto do modelo de conteúdo representa o conteúdo da classe de navegação. Os objetos da classe de navegação não possuem atributos particulares.

As associações possíveis em objetos da classe de navegação são:

- *contentClass*, armazena a lista de classes do modelo de conteúdo que especificam o conteúdo das classes de navegação;
- *menu*, especifica todos os menus acessíveis a partir da classe de navegação em questão;
- *navigationProperty*, que descreve as propriedades de navegação que definem a classe.

Os atributos da classe de navegação são conhecidos como propriedades de navegação e definem o conteúdo da interface de usuário, cujos valores são diretamente tomados das propriedades associadas da classe de conteúdo, ou derivados a partir de uma expressão de seleção.

O atributo das propriedades de navegação é:

- *selectionExpression*, que armazena uma expressão que tem o mesmo tipo da propriedade de navegação e é usada para derivar o valor das instâncias da classe de conteúdo;

A associação relativa às propriedades de navegação é:

- *contentProperty*, que armazena o valor da propriedade de navegação, que é um atributo da classe de conteúdo associada com a classe de navegação.

Um vínculo de navegação é responsável pela conexão de todos os tipos de nós que não sejam classes de processo. Caso a origem ou o destino de um vínculo seja classe de processo, então se usa um vínculo de processo. Vínculos de navegação não possuem atributos ou associações particulares.

Menus são usados para tratar caminhos alternativos de navegação, porém, o menu não é desenhado no modelo de navegação como seria na interface de usuário, pois dois nós que são conectados através de um menu poderiam ser definidos para serem desenhados simultaneamente no modelo de apresentação. Nesse caso, o usuário não deve tomar nenhuma atitude para seguir o caminho de navegação, que seria essencial para um menu em qualquer interface de usuário. Um objeto da meta-classe menu não possui atributos particulares.

A associação relativa a menus é:

- *navigationClass*, que indica a classe de navegação que é origem dos caminhos de navegação definidos no menu.

As primitivas de acesso são usadas para as instâncias de classes de conteúdo que definem o conteúdo das classes de navegação. Não possuem atributos adicionais.

A associação definida nas primitivas de acesso é:

- *accessedAttributes*, que consiste numa lista de propriedades de navegação usados para selecionar a instância da classe de conteúdo.

Índices permitem a seleção de uma instância da classe de conteúdo entre um conjunto das instâncias que foram compiladas durante uma atividade de navegação anterior. Isso significa que um conjunto de objetos da classe de conteúdo é buscado do conjunto anterior no caminho de navegação e é permitido ao usuário escolher um desses objetos. O objeto escolhido se torna objeto de conteúdo para a classe de navegação que sucede o índice no caminho de navegação. Índices não possuem atributos ou associações particulares.

Instâncias da meta-classe consulta são usadas para recuperar o conteúdo de uma fonte de dados. Diferente de índices, consultas não buscam seu conjunto de objetos de classes de conteúdo de um caminho de navegação anterior, e sim de um banco de dados ou qualquer tipo de fonte de dados que suporta consultas. Uma consulta pode precisar de parâmetros de busca e nesse caso, o modelo de apresentação deve ter campos na interface de usuário para preenchimento dos parâmetros solicitados pela consulta. Quando uma consulta não requer parâmetros, ela é executada no momento em que é acessada no grafo de navegação. Consultas não possuem tipos especiais de associação.

O atributo de uma consulta é:

- *filterExpression*, que contém a expressão que define a semântica da consulta.

Um roteiro guiado oferece um processo seqüencial de instâncias de classes de navegação, definindo-se um conjunto ordenado de instâncias de classes de conteúdo como entrada e possui um vínculo de navegação de saída para uma classe de navegação. Não possui associações especiais.

O atributo de um roteiro guiado é:

- *filterExpression*, que tem a mesma sintaxe do filtro de consultas porém, no contexto do roteiro guiado, define a ordem em que instâncias da classe de navegação são visitadas.

3.2.4 Módulo de apresentação

O módulo de apresentação define uma visão abstrata da interface de usuário, tendo como elementos básicos as classes de apresentação, que são baseadas diretamente nos nós do modelo de navegação. As classes de apresentação podem ser acompanhadas de outros elementos através de propriedades de apresentação que têm elementos de apresentação definidos como tipo.

O elemento de apresentação é a superclasse abstrata de todos os elementos do módulo de apresentação, logo, não possui atributos específicos. O elemento de apresentação não possui associações especiais.

A classe de apresentação define a combinação dos elementos de apresentação que exibem o conteúdo de um nó de navegação.

As associações relacionadas àclasse de apresentação são:

- *node*, que indica os nós que serão desenhados pela classe de apresentação;
- *presentationProperty*, especifica a lista de propriedades que constituem a classe de apresentação.

As propriedades de apresentação são usadas para definir o conteúdo das classes de apresentação. O elemento de apresentação que deve ser incluído é usado como o tipo da propriedade de apresentação. Se o elemento contido é de interface de usuário então a propriedade de apresentação pode ser associada com a propriedade de navegação ou de processo que define a localização do dado a ser apresentado ou modificado.

As associações relacionadas apropriedades de apresentação são:

- *presentationElement*, caracteriza o elemento de apresentação que deve ser incluído na classe de apresentação que possui a propriedade de apresentação;

- *navigationProperty*, especifica a propriedade de navegação ou de processo que define a localização do dado que é apresentado ou modificado pelo elemento de apresentação incluso.

A meta-classe página possui a mesma semântica que a classe de apresentação, exceto que a página não pode ser incluída em outra classe de apresentação e não precisam ser associadas a um nó de navegação enquanto a página incluir pelo menos uma classe de apresentação que faça referência ao modelo de navegação. Uma página não possui atributos ou associações adicionais.

Um grupo de apresentação é usado para definir um conjunto de classes de apresentação cujos conteúdos são exibidos alternadamente numa mesma área da página, dependendo da navegação. Se um nó de navegação associado a uma das alternativas é alcançado, o conteúdo dessa classe de apresentação substitui o conteúdo da classe de apresentação atual. Uma das classes de apresentação pode ser definida como padrão, que é automaticamente selecionada se nenhum nó de navegação associado é alcançado. Grupos de apresentação não possuem atributos especiais.

A associação definida nos grupos de apresentaçãoé:

- *default*, que define a classe de apresentação padrão do grupo de apresentação.

O elemento de interface de usuário é a superclasse abstrata responsável por apresentar ou modificar conteúdos. Os elementos de interface de usuário devem ser inseridos em classes de apresentação associadas aos nós de navegação. As subclasses do elemento de interface de usuário podem ser divididas em quatro grupos: o container de interface de usuário, elementos estáticos, entradas de usuário e botões de navegação. O elemento de interface de usuário não define atributos adicionais.

A associação definida no elemento de interface de usuário é:

- *uiContainer*, onde é definido o container de interface de usuário que contém o elemento de interface de usuário.

O container de interface de usuário é uma superclasse abstrata que não é vinculada a nenhum dado diretamente mas pode conter outros elementos de interface de usuário. Um container de interface de usuário não contém atributos adicionais

A associação definida no elemento de interface de usuário é:

- *elements*, que representa a lista de elementos de interface de usuários contidos.

Um formulário tem como função agrupar elementos de interface de usuários que são usados para prover dados para um processo. Não possui elementos ou associações extras definidos.

Uma coleção de âncoras é um container de interface de usuário, mas que apenas comporta âncoras. Pode ser usada para fazer a modelagem da apresentação de um menu ou índice. Não possui atributos adicionais.

A associação definida por uma coleção de âncoras é:

- *anchors*, que representa as âncoras que fazem parte da coleção.

Uma âncora permite ao usuário ativar uma transição em um modelo de navegação através de um vínculo específico. Não possui atributos essenciais.

A associação definida em âncoras é:

- *link*, que representa o vínculo seguido no modelo de navegação quando a âncora é ativada.

Um botão é basicamente um elemento que habilita os usuários a iniciar alguma ação da aplicação *web*. Não possui atributos ou associações próprios.

O elemento de texto é usado para exibir textos estáticos. O conteúdo pode ser fornecido por uma propriedade de navegação. Não possui atributos ou associações próprios.

O elemento imagem é usado para exibir uma imagem estática. O conteúdo fornecido pela propriedade de navegação pode ser interpretado como uma URL especificando a localização de um arquivo de imagem ou diretamente dados de imagem de qualquer formato. Não possui atributos ou associações próprios.

O elemento de entrada de texto consiste em um campo para receber a entrada de texto dos usuários. Não possui atributos ou associações particulares.

O elemento escolha permite selecionar um ou mais valores de um conjunto de possibilidades. Em uma aplicação *web*, existem várias formas para essa funcionalidade ser realizada por elementos HTML concretos. Não possui associações próprias.

O atributo definido no elemento escolha é:

- *multiple*, que define se a escolha pode ser feita através de múltiplos valores.

3.2.5 Módulo de processo

O módulo de processo fornece elementos de modelagem para integração do processo de negócios em um modelo de aplicação *web* UWE, e pode ser dividido em três tarefas:

- Integração do processo de negócios no modelo de navegação, que é habilitada pelas meta-classes, vínculo de processo e classe de processo, que estende os elementos vínculo e nó respectivamente e definem como um processo pode ser alcançado pela navegação e como a navegação irá prosseguir após o mesmo.
- Definição de uma interface de usuário para comportar processos. Processos comumente requerem uma interface de usuário para entrada de dados e apresentação. Essa interface pode ser definida através do modelo de apresentação UWE para cada classe de processo assim como a interface de usuário para navegação. Porém, entradas de usuários são requeridas em diversos pontos do processo. Isso é resolvido criando uma classe de processo para cada passo e associá-las à classe principal do processo integrado ao modelo de navegação. Para cada uma dessas classes, será criada uma classe de apresentação definindo a interface de usuário. Os elementos de interface de usuário são conectados com propriedades de processo da classe de processo correspondente.
- Definição comportamental, que é feita através de uma atividade UML pertencente à classe de processo principal.

As classes de processo são usadas para integrar processos de negócios ao modelo de navegação e definir os dados que serão trocados com o usuário durante o processo. Classes de processos não possuem atributos adicionais.

As associações relacionadas àsclasses de processos são:

- *processActivity*, consiste na atividade UML que define ofluxo do processo;
- *processProperty*, especifica a lista de propriedades que são conectadas ao elemento de interface de usuário e são usadas para definir como a entrada vinda desses elementos é tratada pelo processo.

Vínculos de processos são usados para conectar classes de processo a outros nós de navegação. Não possuem atributos adicionais.

A associação relacionada a vínculos de processos é:

- *processClass*, que é o nó destino do vínculo de processo.

Uma propriedade de processo pertence a um processo e é usada para definir como os dados recuperados de um elemento de interface de usuário são utilizados no fluxo do processo. Propriedades de processo não possuem associações específicas.

O atributo definido nas propriedades de processo é:

- *rangeExpression*, que consiste em uma expressão que pode ser usada para definir a gama dos possíveis valores para a entrada relacionada ao elemento de interface de usuário.

Uma ação do usuário define um ponto em um fluxo do processo quando é solicitado um dado de entrada ao usuário. Quando a ação do usuário é alcançada no fluxo de controle da atividade do processo, os elementos de interface de usuário da classe de apresentação correspondente são exibidos. Depois de enviados os dados, o processo continua. Ações de usuários não possuem atributos adicionais.

A associação definida por ações de usuário é:

- *processClass*, que representa a classe de processo referenciada por uma classe de apresentação e que fornece propriedades de processos que por sua vez são em parte referenciados por elementos de interface de usuário.

4 MODELAGEM DO SISTEMA BIBLIOTECÁRIO *WEB*

Até agora foram vistas as principais características das aplicações *web* e da engenharia *web*. Foi também introduzida uma metodologia para desenvolvimento *web*, a UWE, e foram apresentadas suas principais características e argumentos para sua aplicação na engenharia *web*. No presente capítulo será posto em prática os conhecimentos baseados nos capítulos prévios através da construção de um modelo de aplicação *web*, baseado num sistema de gerenciamento de bibliotecas.

4.1 Definição do módulo de requisitos

A primeira etapa da modelagem é o levantamento de requisitos e a síntese em um diagrama de casos de uso. A seguir será vista uma descrição informal dos requisitos levantados de acordo com as necessidades da aplicação. Em seguida, será sintetizado um diagrama de casos de uso baseado nesses requisitos.

4.1.1 Descrição dos requisitos do sistema

O objetivo principal do sistema é informatizar o gerenciamento e acesso a uma biblioteca universitária e suas unidades setoriais, provendo agilidade e sistematização de controle de livros e outros recursos, e facilitando a pesquisa e operações referentes a situações de usuários.

4.1.2 Diagrama de casos de uso

A figura 4.1 mostra o diagrama de casos de uso do sistema bibliotecário. As linhas pontilhadas nas relações de dependência entre os casos de uso simbolizam extensão.

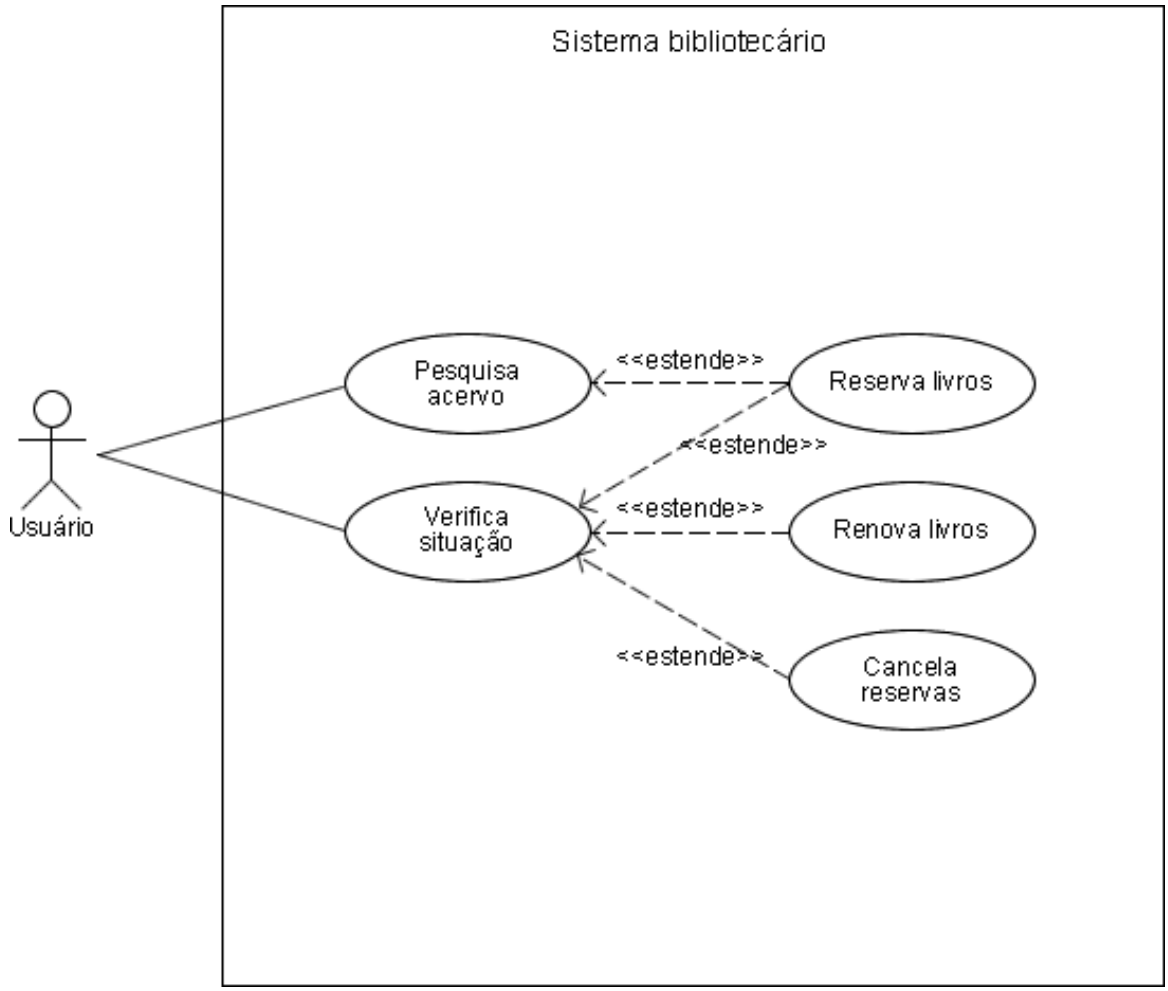

Figura 4.1 – Diagrama de casos de uso do sistema bibliotecário.

4.2 Definição do módulo de conteúdo

O módulo de conteúdo auxilia a visualização da informação relevante do sistema que consiste no conteúdo da aplicação. A representação do conteúdo é feita pelo diagrama de classes UML. No caso presente a informação é fornecida pelas classes livro e setor. A figura 4.2 mostra as classes envolvidas.

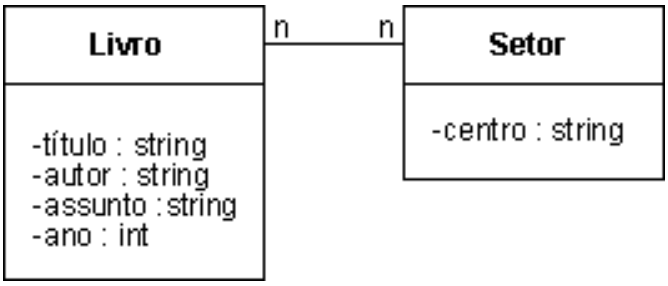

Figura 4.2 – Classes que representam o módulo de conteúdo.

4.3 Definição do módulo de navegação

O modelo de navegação do exemplo demonstra a aplicação dos elementos de navegação definidos anteriormente, e representa a estrutura navegacional da aplicação. O diagrama de navegação pode ser visto na figura 4.3.

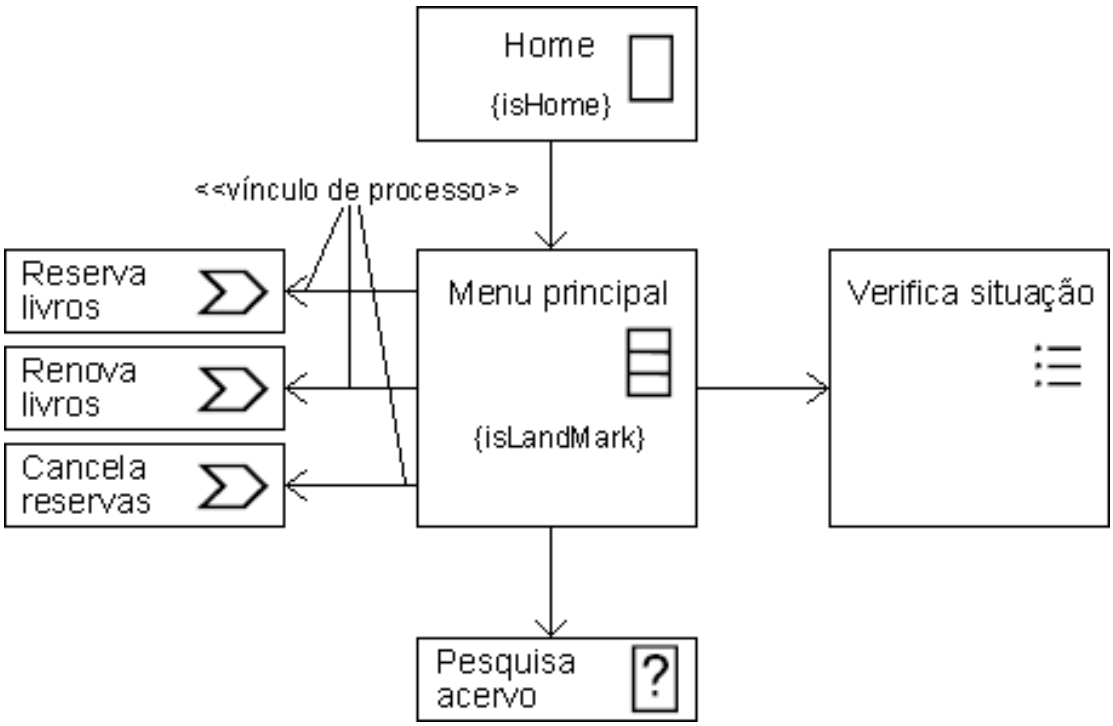

Figura 4.3 – Diagrama de navegação.

4.4 Definição do módulo de apresentação

Finalmente, é feita a análise da apresentação do sistema. O modelo de apresentação usado trata-se de um diagrama composto que define a hierarquia entre os elementos de apresentação, e é mostrado abaixo, na figura 4.4. Esse diagrama apresenta uma abstração da estrutura da interface de usuário, através de uma notação precisa e independente de padrões estéticos, para posteriormente aplicar testes de usabilidade e então finalmente conceber a interface final.

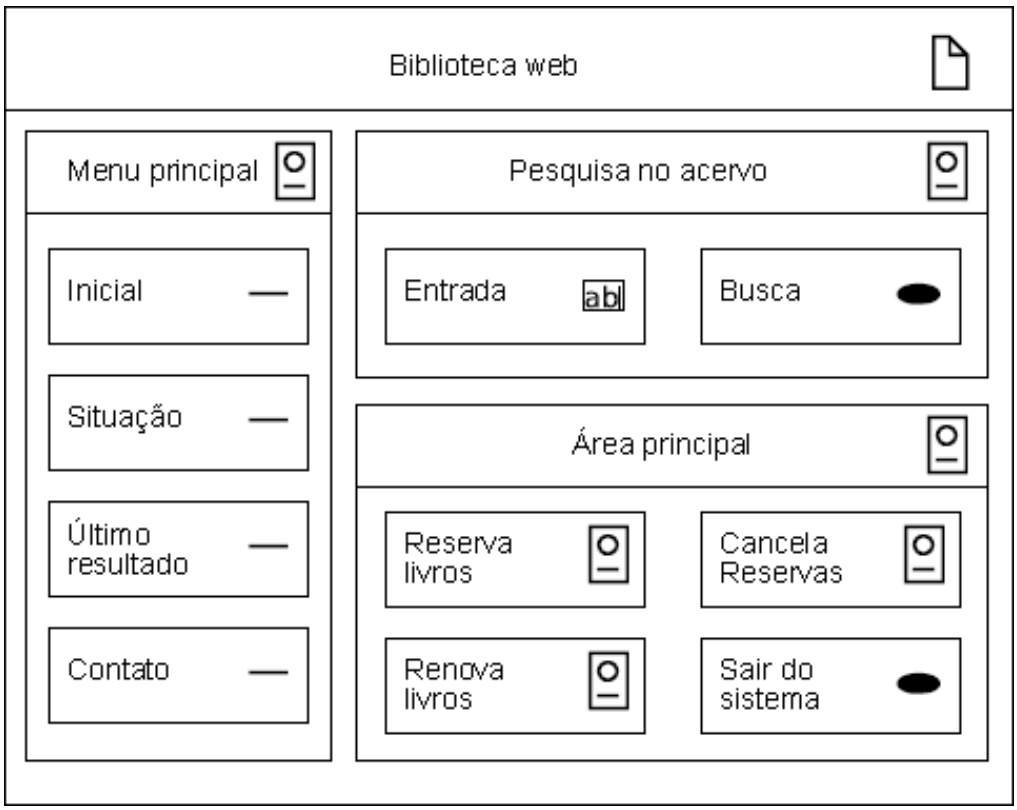

Figura 4.4 – Diagrama do modelo de apresentação.

5 CONCLUSÕES

Através do caminho do desenvolvimento do presente trabalho, foi possível concluir que apesar de a tecnologia avançar em ritmo acelerado e ser suposto que tecnologias se tornam obsoletas da noite para o dia, muitas tecnologias da informação são ainda empregadas apesar das crenças ao contrário. Por causa dessa filosofia de transitoriedade de conhecimentos, comum atualmente, tecnologias revolucionárias não são acompanhadas de metodologias adequadas, fazendo-se assim um mau uso de anos de estudo.

Os métodos de desenvolvimento *web* estão ultrapassando as barreiras da informalidade, fazendo com que processos apropriados sejam estudados e desenvolvidos para a criação de programas para a *internet.*

5.1 Estudos futuros

Esse trabalho serve como base para desenvolvedores ficarem cientes de que existem métodos organizados para desenvolvimento de suas complexas aplicações *web*, para que seus futuros trabalhos sejam mais produtivos e mais fáceis de lidar ao fazer manutenções, atualizações de conteúdo e mudanças de funcionalidades, muito freqüentes nesses tipos de aplicações.

Aos pesquisadores da área de engenharia de *software* e sistemas aplicados à *Internet*, esse trabalho tem uma grande relevância como ponto de partida para conhecer métodos de engenharia *web*, para dessa forma aprimorá-los e usa-los para pesquisas promissoras da área de desenvolvimento *web*, como a *web* semântica.

REFERÊNCIAS BIBLIOGRÁFICAS

BOEHM, B. W. **Software Engineering.** IEEE Transactions on Computers; 1981.

DESPHANDE, Y. **Web Engineering: Creating a Discipline among Disciplines.** IEEE Multimedia; 2001.

DOUGLASS, B. P. **Real-Time UML.** Addison Wesley; 1998.

HAMILTON, K. **Learning UML 2.0.** O'Reilly; 2006.

KAPPEL, G. **Web Engineering The Discipline of Systematic Development of Web Applications.** Heidelberg; 2003.

MURUGESAN, S. **Web Engineering: Introduction and Perspectives.** Sydney; 2005.

PFLEEGER, S. L. **Engenharia de Software: teoria e prática.** Pearson; 2004.

PRESSMAN, R. S. **Engenharia de Software.** Makron Books; 1995.

SELMI, S. S. **Toward a Comprehension View of Web Engineering.** Sydney; 2005.

WINMAN, W. E. **Manual de CGI.** Makron Books; 1997.