UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO - CTC
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA - INE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

ENGENHARIA DE SOFTWARE PARA O DESENVOLVIMENTO DE WEBAPPS E AS METODOLOGIAS OOHDM E WEBML

Márcio Henrique Locatelli

Florianópolis
2003

Márcio Henrique Locatelli

ENGENHARIA DE SOFTWARE PARA O DESENVOLVIMENTO DE WEBAPPS E AS METODOLOGIAS OOHDM E WEBML

Monografia apresentada ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Santa Catarina como requisito parcial para obtenção do título de Especialista em Ciências da Computação.

Orientador: **Prof. Leandro J. Komosinski, Dr.**

Florianópolis

2003

"Não é preciso ter olhos abertos para ver o sol,

nem é preciso ter ouvidos afiados para ouvir o trovão .

Para ser vitorioso você precisa ver o que não está visível"

Sun Tzu

AGRADECIMENTOS

Muitas foram as pessoas que ajudaram na realização deste trabalho. Gostaria de agradecer primeiramente a Deus, por todas as oportunidades; meus pais Sr. Pedro Celso Locatelli e Sra. Clarice Anzolin Locatelli, os quais sempre me incentivaram e ensinaram os melhores caminhos a serem percorridos; a minha irmã Ana Paula Locatelli (Ciências da Computação - UFRGS), que sempre fez críticas construtivas ao meu trabalho ajudando-me a buscar sempre as melhores soluções , a minha amiga Emi Maria que sempre me alegrou nas horas difíceis; a todos meus colegas, em especial: Luiz Carlos de Almeida, Edinardo Potrich e André Forchesatto.

Agradecer especialmente ao Professor e orientador Sr. Leandro J. Komosinski que demonstrou paciência e profissionalismo na orientação, ajudando-me a traçar os caminhos a serem superados, a Professora Patrícia Vilain que encontrou materiais para pesquisa quando foram solicitados, a Unoesc Xanxerê, nas pessoas de seus dirigentes, em especial ao Pró-Reitor de Pesquisa Pós Graduação e Extensão Sr. Nestor Fernandes da Silva e o coordenador da Pós Graduação Sr. Davidson Mazzoco Davi e a todos que direta ou indiretamente contribuíram na elaboração deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

OOHDM	Object Oriented Hypermedia Design Method
UML	Unified Modeling Language
WebApps	Web-based systems and aplications
WWW	World Wide Web
WebML	The Web Modeling Language
CSS	Cascading Style Sheet
XML	Extensible Markup Language
XSL	Extensible Style Sheet Language
HTML	Hypertext Markup Language

RESUMO

Quando eram utilizados somente os meios de comunicação tradicionais (revistas, livros, TV, rádio) para repassar informações e conteúdos, conseguíamos controlar mais facilmente todo o conteúdo que era produzido. Mas este paradigma foi modificado a partir do momento em que novas tecnologias como a Internet e os meios multimídia começaram a surgir no "cenário mundial". Com o surgimento de novos meios de comunicação todo o conteúdo gerado cresceu de maneira impressionante e, em muitos casos, desordenada. Com isso, faz-se necessário o estudo de metodologias que dêem suporte para a geração e manutenção do conteúdo produzido para estas novas tecnologias de comunicação. O presente trabalho descreve a Engenharia de Software para o desenvolvimento de WebApps demonstrando suas principais diferenças com a Engenharia de Software tradicional e as motivações para a utilização da mesma para a análise e construção de WebApps complexos. Apresenta também duas metodologias que auxiliam na Construção de WebApps, sendo elas*: OOHDM – Object Oriented Hypermedia Design Method e WebML – The Web Modeling Language*. Esta monografia foi desenvolvida, inicialmente, com base na crítica a várias aplicações existentes na WEB as quais foram desenvolvidas "*ad-hoc*" (falta de modelos de processo), fazendo com que a dificuldade na manutenção de tais sistemas seja de grande complexidade. Embora a programação para o desenvolvimento de WebApps não seja totalmente diferente da programação de aplicativos tradicionais, a Engenharia de Software para o desenvolvimento de WebApps oferece suporte a partes do projeto que a Engenharia de Software tradicional não prevê.

ABSTRACT

Only when the traditional communication means were used (magazines, books, TV, radio) to pass on information and contents, we got to control the whole content that was produced more easily. But this paradigm was modified starting from the moment in that new technologies as Internet and the multimedia means began to appear in the world scenery. With the new communication means coming out all the generated content increased in an impressive way and disordered, in many cases. Because of that, it is necessary the methodologies study that give support for the generation and maintenance of the produced content for these new communication technologies. The present work describes the Software Engineering for the development of WebApps showing their main differences with the Engineering of traditional Software and the motivations for the use of them for the analysis and construction of complex WebApps. It also presents two methodologies that give support in the Construction of WebApps, they are: OOHDM - Object Oriented Hypermedia Design Method and WebML - The Web Modeling Language. This final paper was developed initially, with base in the critic to several existent applications in the WEB which were developed " ad-hoc " (lack of process models), doing with the difficulty in the maintenance of such systems have great complexity. Although the programming for the development of WebApps not to be totally different from the programming of traditional applications, the Engineering of Software for the development of WebApps offers support to the parts of the project that the Engineering of traditional Software doesn't foresee.

1 - INTRODUÇÃO

A construção de sistemas hipermídia, principalmente sistemas voltados para a Internet, difere do processo de desenvolvimento de software comum, segundo Henicker e Koch[1]: "pessoas com diferentes conhecimentos necessitam trabalhar juntas (designers, webmasters, programadores, pessoas de marketing, comunicação, entre outras..)" tal complexidade na manutenção destas equipes interdisciplinares somada com a complexidade na construção de tais sistemas faz com que o desenvolvimento para a Internet se torne um grande desafio.

Os sistemas desenvolvidos possuem uma grande complexidade, seja no quesito segurança seja no quesito conteúdo, pois as pessoas envolvidas em sua produção, geralmente carecem de conhecimentos básicos de engenharia de software para web, fazendo com que a construção se torne muitas vezes uma tarefa árdua e de tal complexidade com que seja impossíveis administrar. Geralmente os sistemas para Internet começam a ser desenvolvidos sem nenhum planejamento e conforme vão sendo introduzidos conteúdos ou outros sistemas hipermídia vão sendo integrados, a complexidade cresce exponencialmente, fazendo com que somente navegações por ancoras ou elos em hiperlinks não sejam mais eficientes para controlar os projetos.

Para facilitar e administrar a complexidade de construir e manter tais sistemas, podemos utilizar metodologias que dão suporte a modelagem para análise e implementação dos mesmos, fazendo com que o programador do sistema não construa um sistema sem planejamento, pois a modelagem tem como objetivo evitar com que o sistema seja de difícil construção e manutenção.

Nesta pesquisa convencionou-se utilizar o termo WebApps para caracterizar sistemas e aplicativos baseados na web, evitando assim utilizar vários termos, como: engenharia de software para web, engenharia de software para o desenvolvimento de web-sites, engenharia de desenvolvimento de aplicações para Internet, etc... a utilização da sigla WebApps procura caracterizar todos os diferentes tipos de aplicações e sistemas que baseiam-se na World Wide Web.

O objetivo principal desta pesquisa é Apresentar a Engenharia de Software para o Desenvolvimento de WebApps e suas diferenças com a Engenharia de Software tradicional, bem como sugerir duas metodologias para o desenvolvimento de WebApps,

[1] HENNICKER, Rolf; KOCK, Nora. A UML-based Methodology for Hypermedia Design Method.Institute of Computer Science – Ludwig-Maximilians University of Munich, 2000, P. 1.

sendo elas: OOHDM (Object Oriented Hypermedia Design Method) que foi desenvolvida na Pontificia Universidade Católica do Rio de Janeiro, por Gustavo Rossi e Daniel Schwabe. A outra denominada: The Web Modeling Language, desenvolvida no Instituto de Eletrônica e Informação do Politécnico de Milano – Itália.

1.1 – O Problema da Pesquisa

O público alvo desta pesquisa são desenvolvedores de WebApps, professores de engenharia de software e pessoas que se interessam por metodologias que auxiliam a administração da complexidade na construção de WebApps. Esta pesquisa foi motivada por discussões em sala de aula e pelo ambiente de trabalho e também em função de minha necessidade profissional.

Com o avanço e a popularização da World Wide Web, cada vez mais estão surgindo sistemas que se utilizam de tecnologia hipermídia, os quais procuram solucionar qualquer tipo de problema ou oferecer qualquer tipo de produto. A maioria destes sistemas são muito complexos e necessitam de um grande planejamento para o seu desenvolvimento e manutenção, e se as técnicas utilizadas para o planejamento e manutenção forem falhas, o sistema irá tornar-se impossível de manter-se. Para amenizar tais problemas, devem ser utilizados princípios de engenharia de software. Contudo, a engenharia de software tradicional não atende totalmente os requisitos e padrões para aplicações que façam uso da hipermídia, principalmente na World Wide Web, portanto, para solucionarmos tais problemas, faz-se necessária a utilização de uma Engenharia de Software especial que atenda as características destas aplicações.

Diante do exposto considera-se relevante investigar a seguinte problemática:

COMO ADMINISTRAR A COMPLEXIDADE NA CONSTRUÇÃO DE WEBAPPS?

1.2 – Justificativa

Como desenvolvedor de WebApps, desde fevereiro de 1999, acompanho todas as dificuldades no desenvolvimento e manutenção de conteúdo para internet, ficando óbvio que metodologias que dão suporte a análise e implementação de tais sistemas auxiliam muito na criação dos mesmos, também consegui perceber que várias práticas e técnicas existentes apresentam muitas falhas ou são inapropriadas para alguns projetos.

Para facilitar a construção de sistemas e evitar que venham a sofrer toda uma reengenharia posteriormente para seu melhor funcionamento e manutenção, faz-se necessário o estudo de metodologias de suporte e análise.

Como está pesquisa servirá de base para pessoas que desejam obter conhecimentos em engenharia de software para a web, bem como familiarizar-se com metodologias de desenvolvimento, torna-se imprescindível apresentar mais que uma metodologia para as pessoas interessadas efetuarem estudos posteriores.

1.3 – Relevância da Pesquisa

A escolha de uma metodologia para tornar o desenvolvimento de um aplicativo hipermídia para web simplificado e organizado é fator primordial. Por isso, a apresentação de duas metodologias poderá abrir espaço para novas pesquisas que demonstrem também outras metodologias de desenvolvimento, fazendo assim com que saibamos para cada projeto qual será a metodologia que melhor se adaptará, ou seja: se necessitamos uma melhor modelagem navegacional, utilizaremos a metodologia X, mas se necessitarmos uma melhor interface utilizaremos a metodologia Y e assim por diante.

1.4 – Estrutura do Trabalho

O presente trabalho está organizado em quatro capítulos, dispostos conforme a seguinte estruturação:

O primeiro capítulo apresenta a introdução, onde há uma descrição geral do trabalho contextualizando o cenário que levou a escolha do tema, a problemática envolvida, bem como os objetivos e a relevância do assunto abordado .

O segundo capítulo faz uma contextualização histórica da Engenharia de Software tradicional e apresenta de forma geral a Engenharia de Software para o desenvolvimento

de WebApps, demonstrando as características e principais diferenças entre as duas engenharias de software.

No terceiro capítulo são apresentadas e caracterizadas as metodologias que são objeto da pesquisa, também são definidos os elementos que compõem as mesmas, bem como suas características principais através de alguns exemplos de sua utilização.

O quarto capítulo é composto das considerações finais sobre o trabalho desenvolvido, bem como recomendações para futuros trabalhos.

1.5 – Objetivos

1.5.1- Objetivo Geral

O presente trabalho tem como objetivo geral apresentar a Engenharia de Software para o desenvolvimento de WebApps, demonstrando suas características e as principais diferenças entre a mesma e a Engenharia de Software Tradicional, bem como caracterizar duas metodologias de desenvolvimento de WebApps.

1.5.2- Objetivo Específico

- Elaborar uma contextualização histórica da engenharia de software tradicional e a engenharia de software para o desenvolvimento de WebApps.

- Conceituar a engenharia de software para o desenvolvimento de WebApps.

- Caracterizar a metodologia OOHDM (Object Oriented Hypermedia Design Method)

- Caracterizar a metodologia WebML – The Web Modeling Language

2 – CONTEXTUALIZAÇÃO HISTÓRICA DO DESENVOLVIMENTO DE SOFTWARE E ENGENHARIA DE SOFTWARE PARA WEB

2.1 – A Evolução do Software

No início da era da computação a produção de software era considerada uma arte e feita quase de forma artesanal, não existiam métodos para controlar o desenvolvimento e administração das equipes que produziam tais softwares, conforme relata PRESSMAN[2], "A programação de computador era uma arte "secundária" para a qual havia poucos métodos sistemáticos. O desenvolvimento de software era feito virtualmente sem administração[...]." Este método de produção de software fazia os custos e prazo de entrega subirem abruptamente, é lógico afirmar que no inicio o hardware influenciava muito no desenvolvimento dos softwares, devido ao alto custo dos mesmos e o pouco poder de processamento e armazenagem dos dados.

Conforme PRESSMAN[3],

> "Durante as três primeiras décadas da era do computador, o principal desafio era desenvolver um hardware que reduzisse o custo de processamento e armazenagem de dados. Ao longo da década de 1980, avanços na microeletrônica resultaram em maior poder de computação a um custo cada vez mais baixo."

A figura abaixo representa segundo PRESSMAN, a evolução do software:

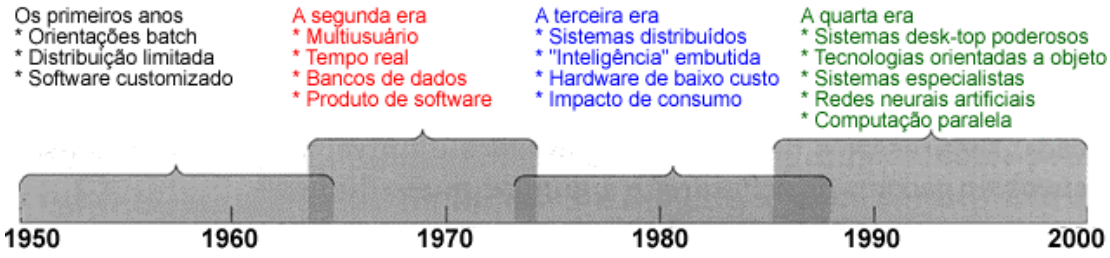

Figura 1 – Evolução do software da década de 50 até os dias de hoje

Fonte: Pressman , Roger S. Engenharia de Software, p. 5.

---

[2] Pressman, Roger S. **Engenharia de Software.** São Paulo; Makron Books, 1995 , p. 5.
[3] Id. 1995, p. 4.

Com o gráfico acima pode constatar-se a evolução na produção durante cinco décadas e as diferentes questões norteadoras de sua produção. Para melhor entender a evolução do software é importante a análise de cada década:

De 1950 a 1960: houve a primeira fase na construção de softwares, os sistemas eram estritamente com orientações batch, a distribuição dos softwares era prejudicada, pois os mesmos eram produzidos sobre medida para cada aplicação. As empresas dependiam muito de seus funcionários(programadores), pois não havia quase documentação alguma e o projeto sempre ficava sobre responsabilidade de alguma pessoa da equipe, é claro que a equipe dependia exclusivamente do conhecimento desta pessoa gestora do projeto, pois a maioria das atividades estavam em sua "cabeça".

Como cita PRESSMAN[4],

> " A maior parte do software era desenvolvida e, em última análise, usada pela própria pessoa ou organização. Você escrevia-o, colocava-o em funcionamento e, se ele falhasse, era você quem o consertava. Uma vez que a rotatividade de empregos era baixa, os gerentes podiam dormir tranqüilos com a certeza de que você estaria lá se defeitos fossem encontrados ."

De 1960 a 1970: aconteceram algumas mudanças significativas no desenvolvimento de software, os sistemas começaram a possuir interatividade através da utilização da multiprogramação e sistemas multiusuários; começaram a surgir sistemas de tempo real, os quais faziam as instruções muito rapidamente produzindo saídas muito rápidas, tal avanço proporcionou o surgimento de sistemas que gerenciam bancos de dados. Outro fato que chama atenção nesta década é o surgimento das "software houses", o software começou a ser desenvolvido para distribuição em larga escala, o que fez com que os sistemas fossem distribuídos para centenas e as vezes milhares de clientes, o que começou a proporcionar um grande problema: a manutenção do software, como administrar a complexidade no atendimento personalizado dos clientes? Como detectar rapidamente falhas e modifica-las? Tais questões trouxeram um enorme problema: **o surgimento da Crise de Software.**

[4] Pressman, Roger S. **Engenharia de Software.** São Paulo; Makron Books, 1995 , p. 5.

Quanto ao surgimento da Crise de software PRESSMANN[5], relata:

> " Uma nuvem negra apareceu no horizonte. Todos esses programas – todas essas instruções – tinham de ser corrigidos quando eram detectadas falhas, alterados conforme as exigências do usuário se modificavam ou adaptados a um novo hardware que fosse comprado. Essas atividades foram chamadas coletivamente de "manutenção de software". O esforço despendido na manutenção de software começou a absorver recursos em índices alarmantes."

De 1970 a 1980: os computadores pessoais começaram realmente a surgir com poder de transformação e processamento compatível com a necessidade dos usuários, o uso de microprocessadores e estações de trabalho aumentaram consideravelmente a produção de bens e serviços; as empresas produtoras de software tem um crescimento espantoso e começam a vender até centenas de milhares de cópias de softwares; o software começa realmente a se diferenciar em questão ao hardware, segundo PRESMANN[6]: "O hardware de computador pessoal está tornando-se rapidamente um produto primário, enquanto o software oferece a característica capaz de diferenciar."

De 1980 a 1990: começou a surgir a quarta era do software, a qual caracteriza-se por sistemas distribuídos, inteligência embutida, hardware de baixo custo e alta do consumo. Vários paradigmas de programação começam a ser "quebrados" e a produção em larga escala e o reaproveitamento de código começam a trazer um grande diferencial entre os programadores utilizando-se de conceitos de programação Orientada a Objeto.

De 1990 aos dias de hoje: sistemas desktop poderosos, tecnologias orientadas a objetos, sistemas especialistas, sólidos sistemas de WebApps, redes neurais artificiais e computação paralela.

---

[5] Pressman, Roger S. **Engenharia de Software.** São Paulo; Makron Books, 1995 , p. 6.
[6] Id. 1995, p. 7

Seja qual for a era da evolução na produção de software que estivermos passando, sempre houveram problemas na administração da complexidade na produção e manutenção dos mesmos, em reposta a estes problemas são utilizadas práticas de engenharia, tal afirmação é confirmada, por PRESSMAN[7],

> Quando se iniciava a década de 1980, uma reportagem de primeira página da revista Business Week apregoava a seguinte manchete: "Software: A Nova Força Propulsora". O software amadurecera – tornara-se um tema de preocupação administrativa. Em meados da década de 1980, uma reportagem de capa de Fortune lamentava "Uma Crescente Defasagem de Software" e, ao final da década, a Business Week avisava os gerentes sobre a "Armadilha do Software – automatizar ou Não". No começo da década de 1990, uma reportagem especial da Newsweek perguntava: "Podemos Confiar em nosso Software?" enquanto o Wall Street Journal relacionava as "dores de parto" de uma grande empresa de software com um artigo de primeira página intitulado "Criar Software Novo: Era Uma Tarefa Agonizante...". Essas manchetes, e muitas outras iguais a elas, eram o anúncio de uma nova compreensão da importância do software de computador – as oportunidades que ele oferece e os perigos que apresenta.

2.2 – Engenharia de Software para o desenvolvimento de WebApps

2.2.1 – Multimídia

Com a popularização da Internet e o grande crescimento da informática, principalmente no meio doméstico e no ensino, cada vez mais aplicações que utilizam multimídia, estão sendo utilizadas para proporcionar repasse de informações de maneira rápida e segura. O termo multimídia caracteriza-se pela utilização de várias mídias e a recuperação de informações através das mesmas. Para BIZOTTO[8], "O termo multimídia tem sido utilizado para designar softwares que utilizam, em conjunto, diversas mídias, como som, texto, imagem e vídeo." Com base em Bizotto, é interessante observar que a televisão possui estas características há muito tempo, mas não é considerada um aparelho que faça uso da multimídia, isto porque a principal característica da multimídia é a interatividade, ou seja: o usuário deve interagir com o meio para obter as informações desejadas, o mesmo participa e cria seu próprio contexto de obtenção das informações.

Com o avanço das aplicações surge a hipermídia, a qual se utiliza de recursos da multimídia e também de hipertextos para disponibilizar informações, conforme

---

[7] Roger S. Pressman, no livro Engenharia de Software, página (3-3).
[8] BIZZOTTO, Eduardo Carlos. **Director 8.0.** São Paulo; Makron Books, 2000 , p.22.

BIZOTTO[9], “Com a evolução da tecnologia, podemos criar links não só em textos mas também em outros tipos de mídias como imagens e vídeos.”

2.2.2 - Hipertextos e Hiperdocumentos

A utilização de hipertextos em hipermídia é muito importante, pois através dos mesmos, pode-se com uma organização não linear da informação, permitir aos seus usuários navegar[10] por um grande aglomerado de documentos contendo informações. Segundo TURINE[11]: “ Um software ou um sistema que apóia a tecnologia hipermídia é denominado sistema hiperdocumento[...] a aplicação, ou o conjunto de documentos on-line gerenciado e manipulado por um sistema hiperdocumento é denominado hiperdocumento.”

Baseado nas idéias de Turine, um documento apresenta uma estrutura linear/seqüencial, enquanto que um hiperdocumento apresenta referências a outros documentos. Em um hiperdocumento são armazenadas as porções de informações e ligações entre elas para as informações completas.

Estrutura básica de um hiperdocumento:

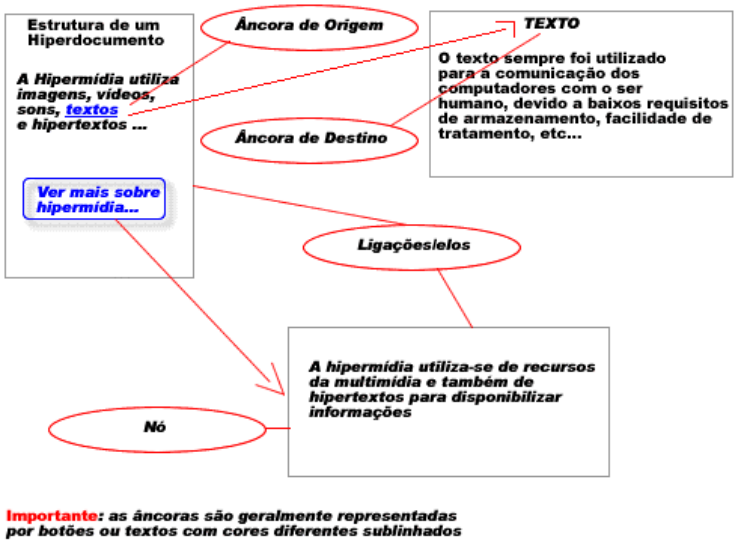

Figura 2 – Representação da Estrutura básica de um hiperdocumento

---

[9] BIZZOTTO, Eduardo Carlos. **Director 8.0.** São Paulo; Makron Books, 2000 , p.22.
[10] Termo bastante utilizado para expressar a leitura em documentos hipermídia
[11] TURINE, Marcelo Augusto Santos. **HMBS- Um Modelo Baseado em Statecharts para a Especificação Formal de Hiperdocumentos.** 1998. 178p. Tese (Doutorado em Ciências da Computação) Universidade de São Paulo / USP, P.2.

Analisando a figura 2 é possível observar alguns elementos essenciais que formam um hiperdocumento; são eles: âncoras de origem e destino, ligações/elos e nós.

a) Nós: é onde se encontra os conteúdos ou informações, segundo TURINE[12]: " [...] nós correspondem às unidades básicas de informação e, normalmente, representam a expressão individual de uma idéia ou um conceito." Também podemos chamar os nós de páginas.

b) Âncoras: podemos classificar as âncoras em dois tipos: âncoras de origem e âncoras de destino. As âncoras de origem possuem ligações as quais são acionadas para chegarmos as âncoras de destino, ou seja, o leitor irá acionar uma das extremidades da âncora para conseguir chegar até o conteúdo que está na âncora de destino.
É importante observar que existe sempre um elo de ligação entre as âncoras e uma âncora sempre estará na extremidade deste elo.

c) Ligações/elos: ligam os nós(páginas). Através destas ligações forma-se uma rede de informações; também através delas o leitor poderá definir seu caminho de navegação pelo hiperdocumento.

Uma forma interessante de representação, para possuirmos uma visão abstrata de um hipertexto, é representá-los como um grafo, segundo SZWARCFITER[13]: "A primeira evidência do uso de grafos data de 1736, quando Euler utilizou-os para resolver um problema de caminhar através de pontes entre duas ilhas cortadas por um rio. Desde então, grafos tem sido utilizados em uma grande variedade de aplicações que vão desde circuitos elétricos até ciências sociais."

---

[12] TURINE, Marcelo Augusto Santos. **HMBS- Um Modelo Baseado em Statecharts para a Especificação Formal de Hiperdocumentos.** 1998 . 178 p. Tese (Doutorado em Ciências da Computação). Universidade de São Paulo / USP, P. 3.
[13] SZWARCFITER, J.L. **Grafos e Algoritmos Computacionais.** São Paulo; Ed. Campus, 1984, P.9.

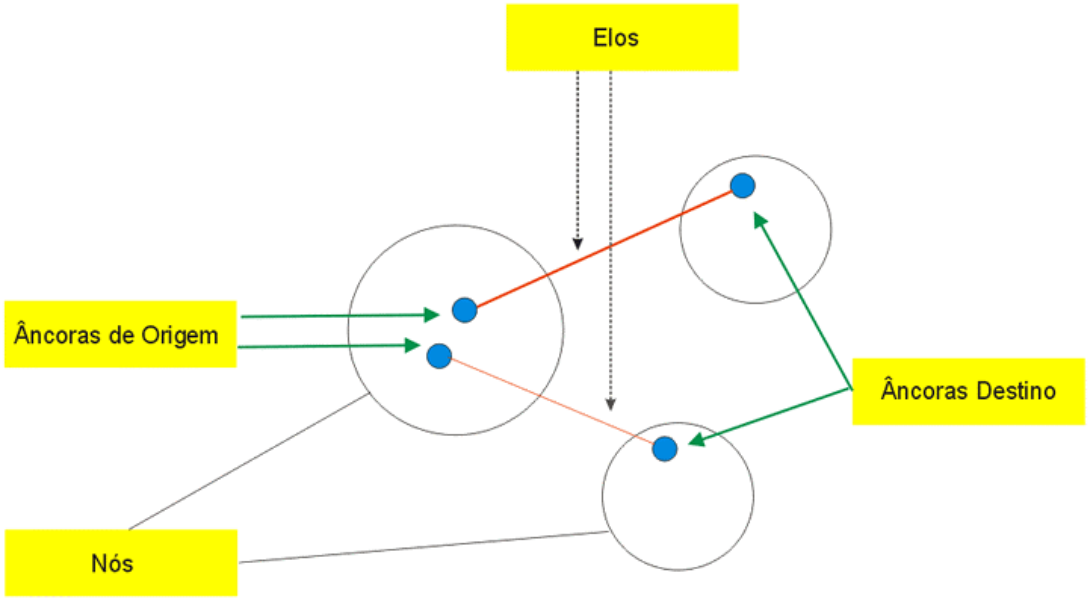

Figura 3 – Representação da Estrutura básica de um hiperdocumento em forma de grafo

2.2.3 - Problemas no desenvolvimento de hiperdocumentos

Através do hiperdocumento um leitor poderá navegar livremente conforme sua necessidade, mas, segundo FRAISSÉ[14]: "[...] navegar eficientemente em um espaço de informação não é simplesmente investigar ou explorar livremente um conjunto desarrumado de ligações ou consultar uma base de dados usando caixas de diálogo ou botões[...]".

Os problemas encontrados no projeto de desenvolvimento de hiperdocumentos são complexos e variados, entre eles pode-se citar:

- Organizar e recuperar as informações em vários domínios da aplicação.
- Tornar simples e acessível o domínio da aplicação com o objetivo de facilitar a navegação do usuário.
- Facilitar a manutenção e a transição de tecnologias (implementação do projeto em outras plataformas)
- Controlar a semântica de navegação.

[14] FRAÏSSÉ, Sylvain : **A Task Driven Design Method and Its Associated Tool for Automatically Generating Hypertexts**. Hypertext 1997.

A representação do hipertexto com sua estrutura básica não fornece um modelo preciso quando há necessidade de modelar um grande aplicativo hipermídia para a web. A grande quantidade de informações e recursos deste sistema necessita de uma engenharia apropriada para sua modelagem para facilitar a navegação e estruturação das informações. Conforme TURINE[15],

> [...] os métodos e as técnicas tradicionais de desenvolvimento de software não são adequados para o projeto de hiperdocumentos, que apresentam requisitos próprios, como a necessidade de gerenciar um grande volume de informações, e de combinar a navegação controlada pelo leitor com a própria natureza das informações multimídia. Tais fatos, motivam a pesquisa por novos modelos e métodos que forneçam diretrizes para gerenciar de maneira sistemática o projeto de hiperdocumentos, ajudando a disciplinar a atividade de autoria, além de encorajar o desenvolvimento de hiperdocumentos estruturados.

2.2.4 - Complexidade no desenvolvimento de WebApps

A complexidade no desenvolvimento de WebApps varia conforme o tipo de projeto realizado, de pequena até larga escala, com grandes sistemas disponíveis na internet e redes corporativas (intranet e extranet).A figura 4 demonstra a crescente complexidade dependendo do tipo de projeto:

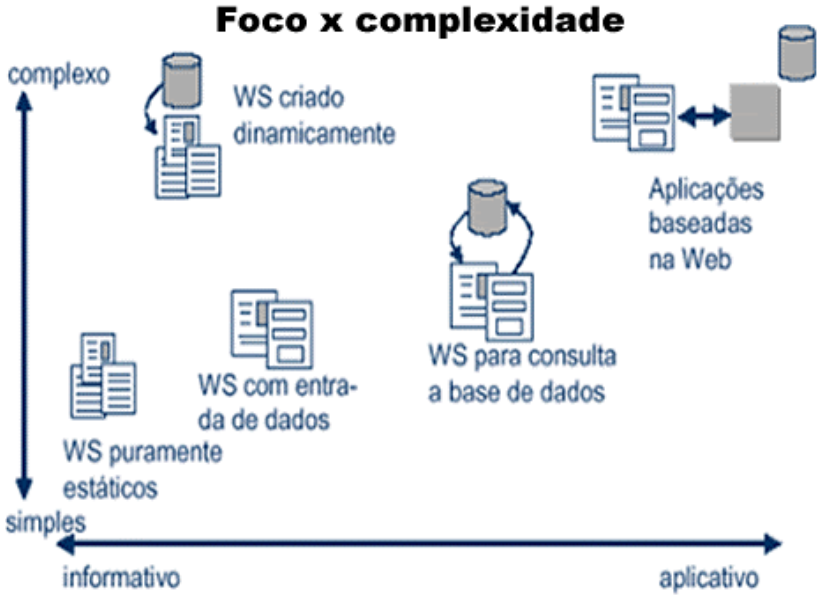

Figura 4 – Foco X Complexidade no desenvolvimento de WebApps

Fonte: Karen Kiomi Nazakato – DCT – UFMS disponível em: http://www.dct.ufms.br/~karen

[15] TURINE, Marcelo Augusto Santos. **HMBS- Um Modelo Baseado em Statecharts para a Especificação Formal de Hiperdocumentos.** 1998 . 178 p. Tese (Doutorado em Ciências da Computação). Universidade de São Paulo / USP, P. 5.

Analisando o gráfico observa-se que o nível de complexidade é pequeno caso a aplicação desenvolvida seja somente de caráter informativo estático. Neste caso o projeto deverá preocupar-se somente com a navegação do usuário e a disponibilização de informações. Mas conforme o projeto começa a ter uma interação maior com sistemas de banco de dados, informações geradas dinamicamente e troca de informações com outros sistemas, a complexidade vai crescendo exponencialmente.

Para facilitar a análise do tipo de projeto a ser construído, torna-se interessante, dividir as WebApps em algumas categorias, sendo que uma WebApps poderá pertencer a mais de uma categoria. A tabela 1 demonstra as várias categorias em que as WebApps podem ser agrupadas:

Categoria	Exemplos
Informacional	Jornais e Revistas On line, catálogos de produtos, classificados on line, livros on line.
Interativos	Formulários de registro, apresentação de informações customizadas, jogos on line
Transacional	Lojas eletrônicas, oferecimento de bens e serviços, bancos on line.
Workflow	sistemas de gerência, planejamento, acompanhamento online, controle de mercadorias.
Ferramentas colaborativas	Sistemas diversos de autoria.
Comunidades on line, lugares de negócios	Salas de bate-papo, sistemas que recomendam produtos ou serviços, e- marketplaces
Portais	Compras On Line, fornecedores de conteúdo

Tabela 1 – Categoria de WebApps

Fonte: GINIGE, A.; MURUGESAN, S. **Web Engineering: An Introduction,** P. 14.

Um dos grandes problemas que influencia na manutenção das aplicações demonstradas na tabela acima, é a interdisciplinaridade das equipes envolvidas nos projetos e o grande número de categorias disponíveis para a construção de WebApps.

Segundo Koch[16],

> The development of hypermedia systems differs from the developing process of traditional software in several dimensions. People with different skills are involved in the process, such as authors, layout designers, programmers, multimedia experts and marketing specialists. The role of the user is augmented and makes it more difficult to capture the requirements of the application. The non-linearity of the hyperdocuments as well as the possibility to connect easily to other hypermedia applications increments the complexity and risk of "lost in the hyperspace.

Do exposto pode-se constatar que é realmente difícil contornar os problemas enfrentados ao deparar-se com pessoas de diferentes formações e conhecimentos envolvidas no processo de desenvolvimento de WebApps. Com tal dificuldade soma-se o contexto de navegação de tais sistemas, pois o usuário é capaz de construir seu próprio contexto, aumentando muito a possibilidade do sistema apresentar falhas.

Outro fator problemático no desenvolvimento são os aspectos estéticos e cognitivos. Para NANARD[17]: "[..] engineering has to take into account aesthetic and cognitive aspects as well traditional software engineering environments do not support." Realmente tais aspectos diferenciam as aplicações, pois o conteúdo visto por um usuário poderá ser totalmente diferente do conteúdo visto por outro se estiver utilizando configurações ou browser diferentes. Tal complexidade aumenta se o usuário também possuir dificuldades cognitivas.

---

[16] HENNICKER, Rolf; KOCK, Nora. **A UML-based Methodology for Hypermedia Design Method**. Institute of Computer Science – Ludwig-Maximilians University of Munich, 2000, P.1.

[17] Nanard J., Nanard M. **Hypertext design environments and the hypertext design process**. Communication of the ACM, August 1995, Vol 38 (8), (1995) 49-56.

2.3 - Engenharia de Software para o desenvolvimento de WebApps.

Quando falamos em Engenharia de software, devemos ter claro que a mesma destina-se a aplicar métodos e tecnologias da Engenharia para planejar, especificar, desenhar, implementar, validar, testar, medir, manter e melhorar os softwares. A engenharia de software para WebApps pode ser definida segundo GRAEF[18]: como: "Aplicação de um enfoque sistemático, disciplinado e quantificável para o desenvolvimento e evolução de aplicações Web, com alta qualidade e a um custo efetivo".

A engenharia de software para WebApps é realmente importante, pois auxilia em todo o processo de desenvolvimento em todas as áreas que se envolvem no projeto: hipermídia, banco de dados, design, artes, comunicação, computação gráfica, linguagens computacionais, etc.

2.4 - Objetivos da Engenharia de software para WebApps

Desde a famosa "Crise de Software" ocorrida a trinta anos atrás, os programadores buscam soluções que facilitem o processo de criação e manutenção de sistemas complexos, com o surgimento de aplicações Hipermídia tanto em cd-rom como para Web e sua popularização, vários paradigmas de programação já tiveram que ser mudados, mas isso não afirma que a programação para a construção de WebApps seja muito diferente da programação tradicional, porém devem ser tomados certos cuidados ao planejar tais sistemas.

O principal objetivo da Engenharia de software para web é desenvolver WebApps com grande eficiência e qualidade, devido ao fato que na maioria das vezes os aplicativos hipermídia para Web são desenvolvidos partindo-se de pequenas aplicações e migrando-se para grandes aplicações, fazendo com que as mesmas sejam difíceis de manter-se.

---

[18] GRAEF, G.; GAEDKE, G. Development and Evolution of Web-Applications using the WebComposition Process Model. International WorkShop on Web Engineering at 9th International, World Wide Web Conference (WWW9), Amsterdam , Holanda, maio, 2000.

Dentre outros objetivos da engenharia de software para o desenvolvimento de WebApps pode-se citar:

- Evitar o desenvolvimento "*ad hoc*" (falta de modelos de processo).
- Garantir o acompanhamento das evoluções tecnológicas
- Garantir o cumprimento de prazos através de estimativas de métricas precisas.
- Gerenciar equipes interdisciplinares, facilitando a comunicação entre as diferentes áreas envolvidas no processo.
- Diminuir custos de criação e manutenção.
- Documentar todas as fases do projeto.
- Proporcionar uma navegação entre as informações de forma intuitiva e organizada.

2.4 - Mitos da Engenharia de software para o desenvolvimento de WebApps

Desde o ínicio do desenvolvimento de WebApps existem alguns mitos que proporcionam muitas vezes erros em projetos e conseqüentemente a perca da qualidade dos sistemas/aplicativos desenvolvidos.

Mito 1 – O desenvolvimento para Web é artesanal.

Realidade: é claro que o desenvolvimento para web exige arte, mas não se deve confundir arte com não planejamento. A arte ou criação do design visual de um projeto é apenas um passo no desenvolvimento total, e os criadores ou desenvolvedores do projeto necessitam de metodologias de modelo de processos que lhes ofereçam suporte na criação.

Mito 2 – O desenvolvimento pode ser "*ad hoc*".

Realidade: o estilo solitário de desenvolvimento não pode ser utilizado no desenvolvimento de um projeto de um WebApps com qualidade. Os modelos de processo devem estar bem definidos, garantindo-se assim a manutenção da complexidade na interdisciplinaridade de todas as equipes envolvidas no projeto.

Mito 3 – Uma vez desenvolvido o projeto ele está totalmente pronto.

Realidade: um WebApps está em constante desenvolvimento, as tecnologias modificam-se muito rapidamente, a necessidade de manutenção é muito grande e o modelo de processo adotado deve prever modificações constantes para melhora de performance, disponibilização de conteúdo, etc...

Mito 4 – O custo de desenvolvimento pode ser calculado sem métricas que garantam estimativas e prazos, conforme vai desenvolvendo-se o projeto vai disponiblizando-se conteúdo.

Realidade: segundo o Cutter Consortium[19], 79% de todos os websites atrasam seu cronograma e 63% excedem seu orçamento. Tais dados confirmam que estimativas e métricas para os cálculos de entrega e custos devem fazer parte do projeto de desenvolvimento.

Mito 5 – Qualquer desenvolvedor está apto a planejar e construir WebApps.

Realidade: o desenvolvimento de WebApps não é no estilo "conwboy" (solitário), pois, várias pessoas de diferentes áreas devem se envolver no projeto, além disso o desenvolvedor deverá criar uma cultura diferente no seu estilo de planejamento e programação.

Mito 6 – Após o desenvolvimento todos os usuários irão acessar o sistema/aplicativo igualmente.

Realidade: além de problemas visuais, auditivos e cognitivos, ainda deve ser levado em conta a diferença de sistemas operacionais e browsers utilizados pelas diferentes pessoas que acessam o sistema/aplicativo.

Mito 7 – A engenharia de software tradicional atende todos os requisitos para o desenvolvimento de webapps de qualidade.

[19] Fonte dos dados: Cutter Consortium, dados de novembro de 2000.

Realidade: a engenharia de software para o desenvolvimento de WebApps não é idêntica a engenharia de software tradicional. Portanto várias técnicas utilizadas na engenharia de software tradicional não se aplicam para a engenharia de software para o desenvolvimento de WebApps.

2.5 - Engenharia de Software Tradicional X Engenharia de Software para o desenvolvimento de WebApps

Tanto a Engenharia de Software tradicional quanto a Engenharia de software para o desenvolvimento de WebApps, preocupam-se em facilitar o desenvolvimento e manutenção de produtos de software complexos, aplicando para isso métodos, técnicas, ferramentas case, e modelos específicos. Todas essas metas e técnicas utilizadas visam gerenciar a qualidade de produção, prazos e pessoas envolvidas no projeto. Mesmo que as duas engenharias tenham o mesmo objetivo e que a Engenharia de Software para o desenvolvimento de WebApps utilize alguns princípios da Engenharia Tradicional, não é possível utilizar as mesmas técnicas da Engenharia de Software Tradicional na Engenharia de Software para o desenvolvimento de WebApps, ou seja, a Engenharia de Software para o desenvolvimento de WebApps não é uma cópia da Engenharia de Software Tradicional, como muitos profissionais pensam.

2.5.1 - Diferenças entre Engenharia de Software Tradicional e Engenharia de Software para WebApps

a) Ciclo de vida do produto desenvolvido

Uma das principais diferenças entre as duas Engenharias é o ciclo de vida do produto a ser desenvolvido, a evolução do software tradicional é muitas vezes menor que a evolução de um WebApps, devido ao caráter tecnológico e informacional de um WebApps evoluir de maneira constante. O que influi realmente no ciclo de vida de um WebApps é seu desenvolvimento ser uma mistura de arte e programação, entre marketing e computação, entre relações internas e externas, enfim a multidisciplinaridade envolvida na Engenharia de Software para WebApps é extremamente envolvente e desafiadora.

Como cita GINIGE[20],

> Web-based systems change and grow rapidly in their requirements, contents, and functionality during their life cycle—much more than what we'd normally encounter in traditional software, information, and engineering systems. Web-basedsystem development is a continuous activity with-out specific releases as with conventional soft-ware. Thus, a Web-based system is like a garden—it continues to evolve and grow.

Baseado nas idéias de Ginige observa-se realmente que um WebApps pode ser comparado com um jardim que está sempre crescendo, pois além de um WebApps fornecer conteúdo que podem ser informações ou serviços, tais conteúdos necessitam mudanças rápidas e contínuas para cumprirem o propósito a que se destinam.

b) Público alvo

A abrangência de uma aplicação convencional possui um público muitas vezes definido de usuários, público esse que pode ser controlado, enquanto que no desenvolvimento de um WebApps a classe de usuários pode ser muito maior, as vezes sendo impossível saber exatamente o perfil de todos os usuários que irão interagir com o sistema a ser construído. A tabela abaixo demonstra as diferenças entre os públicos nas duas engenharias:

Engenharia de Software Tradicional	Engenharia de Sofware para Des. De WebApps
• Público bem definido	• Público diversificado

Tabela 2 – Público da Engenharia de Software

c) Tecnologias Utilizadas

As tecnologias utilizadas na produção de software tradicional aplicando técnicas de engenharia de software são mais estáveis, isto decorre do fato que o desenvolvimento deste tipo de software ocorre a muito mais tempo, além disso existem excelentes ferramentas disponíveis para análise, projeto, implementação e testes. Já no desenvolvimento de

[20] GINIGE, A.; MURUGESAN, S. **Web Engineering: An Introduction.** University of Western Sydney, Australia, IEEE, 2000, P. 16.

WebApps as tecnologias estão em constante evolução, embora existam diversas ferramentas a nível de implementação, existem poucas para análise, projetos e outras atividades para o desenvolvimento.

3 – APRESENTAÇÃO DAS METODOLOGIAS

3.1 – A Metodologia OOHDM (Object Oriented Hypermedia Design Method)

O método OOHDM auxilia na construção de aplicações hipermídia em larga escala, estas aplicações podem ser aplicações para CD-ROM, quiosques, desenvolvimento de WebApps, etc...

O método OOHDM é um precursor do modelo HDM (o modelo HDM é o primeiro modelo que se conhece para o projeto de aplicativos hipermídia). Para consolidar-se na área de modelagem de aplicações hipermídia, o OOHDM herdou de seu precursor algumas idéias e enriqueceu outras, como é possível observar na tabela abaixo.

Características Herdadas do HDM	Características Criadas ou Enriquecidas
Reconhecimento de que a modelagem é independente do ambiente e do modelo de referência.	Não é apenas uma abordagem de modelagem, mas sim uma metodologia, pois aborda muitas atividades.
Reforça as estruturas hierárquicas, oferecendo a possibilidade da construção de agregados.	Na fase de modelagem conceitual as primitivas são mais ricas. Como são orientada a objetos suportam: agregação, generalização e herança.
Inclui o conceito de perspectiva de atributo, onde cada classe folha implementa uma visão possível de atributo.	É possível fazer uso do esquema conceitual para definir perfis de usuários diferentes.

Tabela 3 – Características herdadas e criadas no OOHDM

Fonte: The object Oriented Hypermedia Design Method

disponível em: http://www.telemidia.puc-rio.br/oohdm/oohdm.html

O OOHDM trata o processo de desenvolvimento de um WebApps em quatro etapas diferentes:

- Modelagem Conceitual;
- Modelagem Navegacional;
- Projeto de Interface abstrata;
- Implementação.

Segundo Schwab et al[21],

> “ As atividades são executadas em uma mistura de desenvolvimento, interatividade e prototipação, baseada na construção de modelos. Durante cada atividade um grupo de modelos de objetos descrevendo cada projeto em particular são construídos ou enriquecidos com interações.

Baseado na informação de Schwabe, observa-se claramente que a metodologia OOHDM faz com que as atividades sejam executadas iterativamente, isto é bastante importante, pois, é possível ir realizando uma atividade em paralelo com outra e retornarmos para qualquer atividade caso for necessário.

A figura abaixo, demonstra um esboço dos modelos e atividades em OOHDM:

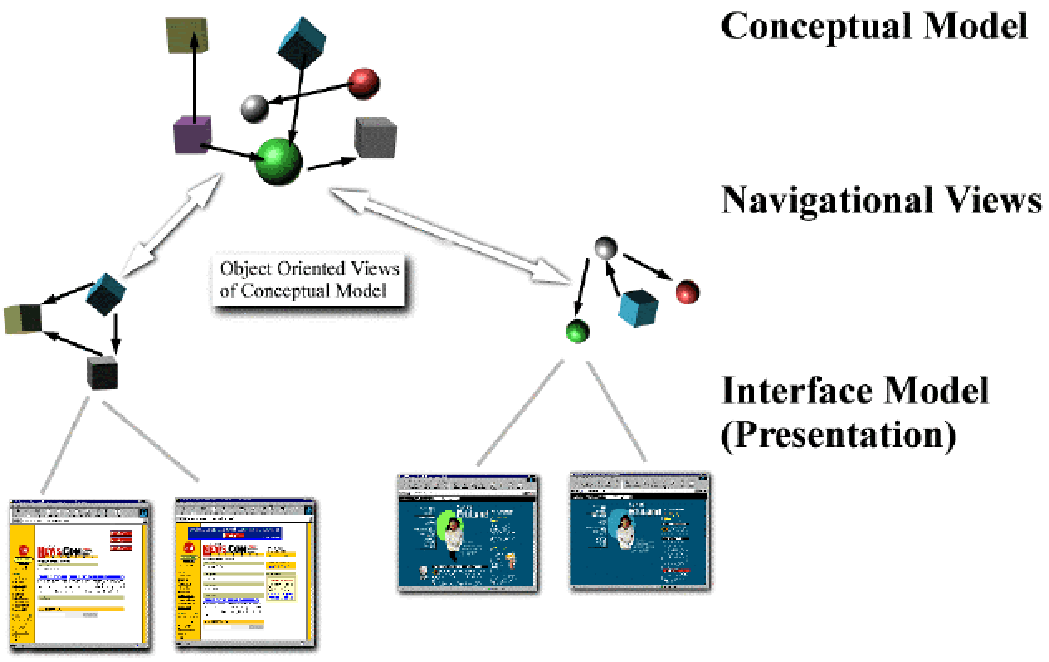

Figura 5 - Esboço e modelo das atividades em OOHDM.

Fonte:Schwabe et al: An Environment for Implementation of Hypermedia Application in the WWW, p. 2

[21] SCHWABE , Daniel; ALMEIDA, Rita; MOURA, Isabela. **OOHDM-Web: An Environment for Implementation of Hypermedia Application in the WWW**. SigWEB Newsletter, Vol. 8, #2, Junho de 1999.

Utilizando-se a metodologia OOHDM é possível desenvolver projetos modulares e de fácil manutenção, pois a modelagem conceitual, navegacional e o projeto de interface são tratados como atividades separadas o que nos permite concentrar-se em diferentes interesses, resolvendo um a um e retornando para cada etapa construída quando for necessário, com isso obtemos também uma estrutura para o raciocínio sobre o processo de todo o projeto, obtendo experiência específica para cada atividade. OOHDM também oferece independência na escolha de linguagens e ambientes de programação fazendo com que a escolha do projeto seja mais ampla pois pode-se tanto trabalhar com o paradigma de orientação a objetos como programação estruturada, ou as duas ao mesmo tempo.

Segundo Schwab et al[22],

> Primitivas do projeto podem ser diretamente mapeadas em linguagens de implementação ou ambientes não orientados a objeto (tais como: HTML ou Toolbook). Conseqüentemente, OOHDM pode ser usado em linguagens e ambientes puramente orientados a objeto ou ambientes híbridos (como os usualmente encontrados na Internet).

3.1.1 – Modelagem Conceitual

O ponto chave da Modelagem Conceitual é analisar o domínio da aplicação, obtendo todas as informações importantes para o desenvolvimento do projeto, isso não quer dizer que todas essas informações serão utilizadas na implementação, pois partes dela poderão ser descartadas. Segundo Schwabe e Vilain [23]: "O resultado da modelagem Conceitual de uma aplicação consiste de um esquema conceitual contendo os objetos do domínio da aplicação (classes, relacionamentos e subsistemas)." Para a construção dos objetos do domínio da aplicação é utilizada uma notação baseada em UML.

---

[22] SCHWABE , Daniel; ALMEIDA, Rita; MOURA, Isabela. **OOHDM-Web: An Environment for Implementation of Hypermedia Application in the WWW**. SigWEB Newsletter, Vol. 8, #2, Junho de 1999, P. 2.

[23] SCHWABE , Daniel; VILAIN, Patricia. **Notação da Metodologia OOHDM.** PUC-RIO, abril 1999, P. 4.

A figura abaixo representa duas classes com múltiplos atributos e sua relação:

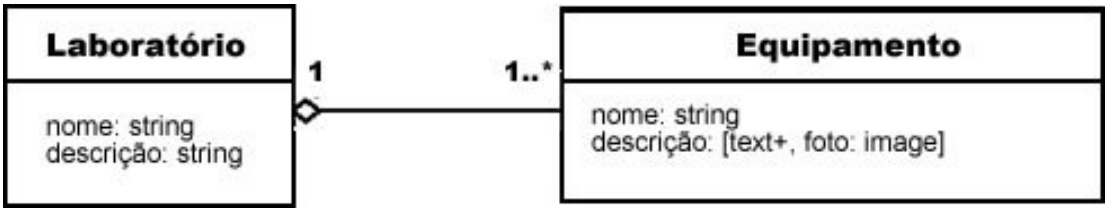

Figura 6 – Classes em OOHDM

Na análise da figura acima é importante observar alguns pontos importantes:

a) Classes

As classes em OOHDM são iguais as classes utilizadas em modelagens Orientada a Objetos. Conforme Schwabe e Vilain[24]: "As classes representam um conjunto de entidades que apresentam as mesmas características (atributos, relacionamentos e métodos)."

b) Relacionamentos

Os relacionamentos fazem a ligação entre objetos, são representados através de uma linha que conectam uma ou mais classes. No exemplo acima é possível observar um relacionamento binário entre a classe laboratório e equipamento. Os relacionamentos podem ser: unário, binário e ternário.

c) A cardinalidade

Em OOHDM a cardinalidade dos relacionamentos é representada por um subconjunto de inteiros não negativos, da seguinte maneira:

0 .. 1 - a cardinalidade pode ser 0 ou 1;
1 - a cardinalidade é igual a 1;
0 .. * - a cardinalidade pode variar de 0 a infinito

[24] SCHWABE , Daniel; VILAIN, Patricia. **Notação da Metodologia OOHDM.** PUC-RIO, abril 1999, P. 5.

* - a cardinalidade pode variar de 0 a infinito

1 .. * - a cardinalidade pode variar de 1 a infinito

1 .. 6 - a cardinalidade pode variar de 1 até 6

Na figura acima existe uma cardinalidade de 1 para n (1 até infinito), a qual demonstra que 1 laboratório possui de 1 até n equipamentos e que os equipamentos estão em exatamente 1 laboratório.

d) Agregação

As classes no esquema conceitual podem ser construídas usando hierarquias de agregação, generalização/especialização.

Na figura 6 utilizada como exemplo, observa-se claramente que o objeto laboratório é composto por vários equipamentos, segundo Schwabe e Vilain[25]: "A agregação é representada por um relacionamento com um losango vazio no lado da classe que agrega as outras."

e) Atributos

Os atributos das classes representam as propriedades dos objetos. OOHDM permite que cada perspectiva represente um valor diferente para cada atributo. Para representar múltiplas perspectivas em um atributo, as mesmas deverão ficar entre colchetes "[ ]", sendo que cada uma deverá ser representada por um identificador , seguido do caracter ":" e de um tipo.

Na figura 6 utilizada como exemplo observa-se que o *atributo descrição* da classe equipamento apresenta duas perspectivas: um texto para sua descrição e uma foto para sua representação:

Descrição: ***[text+, foto:image]***

[25] SCHWABE , Daniel; VILAIN, Patricia. **Notação da Metodologia OOHDM.** PUC-RIO, abril 1999, P. 15.

Segundo Schwabe e Vilain[26],

> Existem duas situações em que são usadas múltiplas perspectivas. A primeira situação é quando o atributo com múltiplas perspectivas é apresentado no mínimo em uma mesma perspectiva sempre que o objeto é mostrado. Nesta situação, é necessário que uma das perspectivas seja default, sendo assinalada com o caracter "+" A perspectiva default representa a perspectiva na qual o atributo será apresentado em todas as instâncias. A perspectiva default é a única perspectiva que não precisa apresentar um identificador, neste caso recebendo o identificador do atributo.

Para representarmos todas as classes do domínio da aplicação, bem como seus relacionamentos utilizamos o chamado: **Esquema Conceitual.** Para construir-se o esquema conceitual de classes não é necessário utilizar qualquer método em particular, ou seja: metodologias conhecidas para o mapeamento de classes poderão ser utilizadas, um exemplo poderia ser a OMT, ou outra utilizada em larga escala.

A figura abaixo demonstra um exemplo de um esquema conceitual de um site sobre arquitetura, nesta figura pode-se observar as perspectivas de atributos com múltiplos valores, os relacionamentos entre as classes, sua cardinalidade e a identificação das classes do esquema.

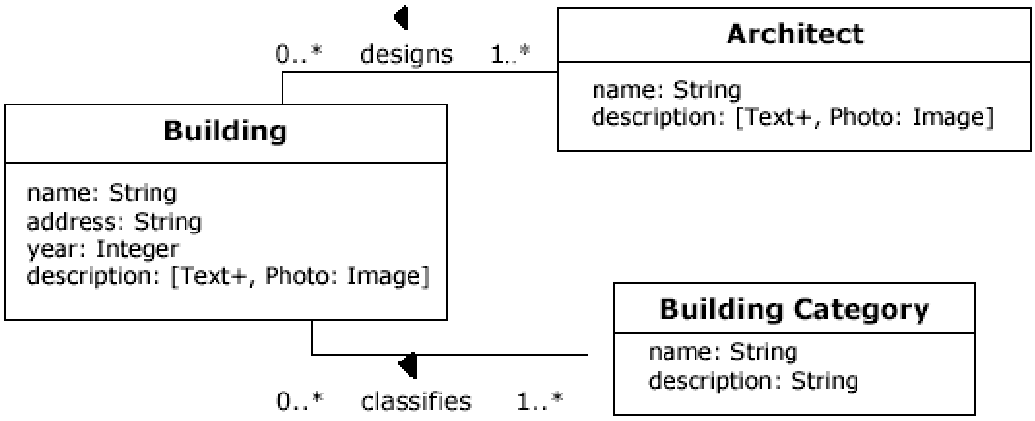

Figura 7 – Esquema Conceitual

Fonte: Schwabe et al , OOHDM-WEB, p. 3

[26] SCHWABE , Daniel; VILAIN, Patricia. **Notação da Metodologia OOHDM.** PUC-RIO, abril 1999, P.7.

3.1.2 – Modelagem Navegacional

Um dos principais desafios de um WebApps é definir as informações que irão fazer parte da aplicação, bem como a navegação entre as mesmas. Em OOHDM podemos definir a modelagem navegacional a partir da modelagem conceitual ou diretamente (sem que a modelagem conceitual tenha sido executada). Geralmente quando é feita uma reengenharia de um aplicativo já existente a modelagem navegacional é definida a partir da modelagem conceitual ou quando existem vários perfis de usuários para uma mesma aplicação.

Uma das principais inovações do OOHDM é realmente reconhecer que os objetos navegacionais são diferentes dos objetos conceituais, pois os objetos em que o usuário navega não são os objetos do modelo conceitual, mas sim outros tipos de objetos os quais são construídos de um ou mais objetos conceituais.

3.1.2.1 – Esquema de classes navegacionais

No esquema de classes navegacionais são definidos os conjuntos de nós e elos que abrangem o contexto navegacional da aplicação. Dependendo do tipo e do escopo da aplicação poderão existir vários esquemas navegacionais (um para cada tipo de visão do sistema).

Quanto a representação do esquema de classes navegacionais, Schwabe e Vilain[27] relatam que:

> "O esquema navegacional é um diagrama onde os nós são representados por retângulos e os elos por linhas, assim como no esquema conceitual. Entretanto, as linhas que representam os elos são direcionadas representando a possível navegação entre as informações disponíveis."

3.1.2.1.1 - Nós

Os nós representam um conjunto de instâncias que apresentam as mesmas características (atributos, elos e métodos). Um nó provém de uma classe, podendo também ser organizado utilizando-se de hierarquias de generalização/especialização.

---

[27] SCHWABE , Daniel; VILAIN, Patricia. **Notação da Metodologia OOHDM.** PUC-RIO, abril 1999, P. 24.

Conforme Schwab et al[28]:

> Consequentemente, atributos dos nós são definidos como "imagens" de objetos de classes conceituais. Isto significa que um nó pode ser definido para providenciar acesso a atributos de diferentes classes relacionadas no esquema conceitual. Esta "imagem" é orientada para uma certa classe de usuários e suas respectivas tarefas.

A figura abaixo representa um nó (para diferenciar um nó de uma classe conceitual, inclui-se um linha vertical na primeira parte a direita).

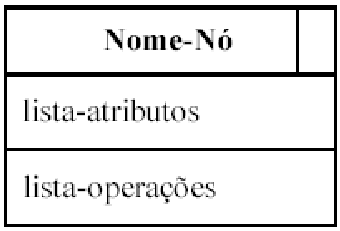

Figura 8 – Representação de um nó em OOHDM

Fonte: SCHWABE , Daniel; VILAIN, Patricia. **Notação da Metodologia OOHDM**, p. 25

Para mapearmos objetos agregados do esquema conceitual podemos utilizar um nó composto, fazendo com que a complexidade da especificação seja drasticamente reduzida. Assim, simplificando a definição da semântica da navegação entre os nós.

Para exemplificar, a figura abaixo utiliza-se de um exemplo clássico de agregação (já demonstrado na figura número 6) entre um laboratório e seus equipamentos:

[28] SCHWABE , Daniel; ALMEIDA, Rita; MOURA, Isabela. **OOHDM-Web: An Environment for Implementation of Hypermedia Application in the WWW**. SigWEB Newsletter, Vol. 8, #2, Junho de 1999.

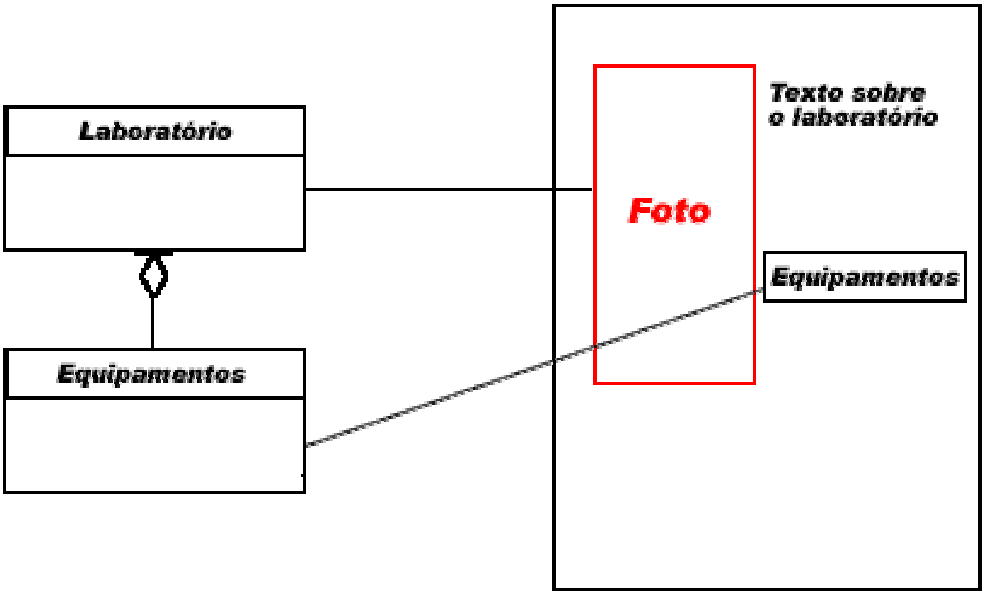

Figura 9 – Representação de um nó composto

Com a figura acima percebe-se que a navegação entre um nó composto deve permitir ao usuário mover-se no interior dos nós de maneira coerente, fazendo com que a navegação torne-se realmente intuitiva.

3.1.2.1.2 – Representando o esquema de classes navegacionais

A figura abaixo representa o esquema de classes navegacionais para o exemplo da arquitetura utilizado no esquema conceitual (conforme demonstrado na figura número 7).

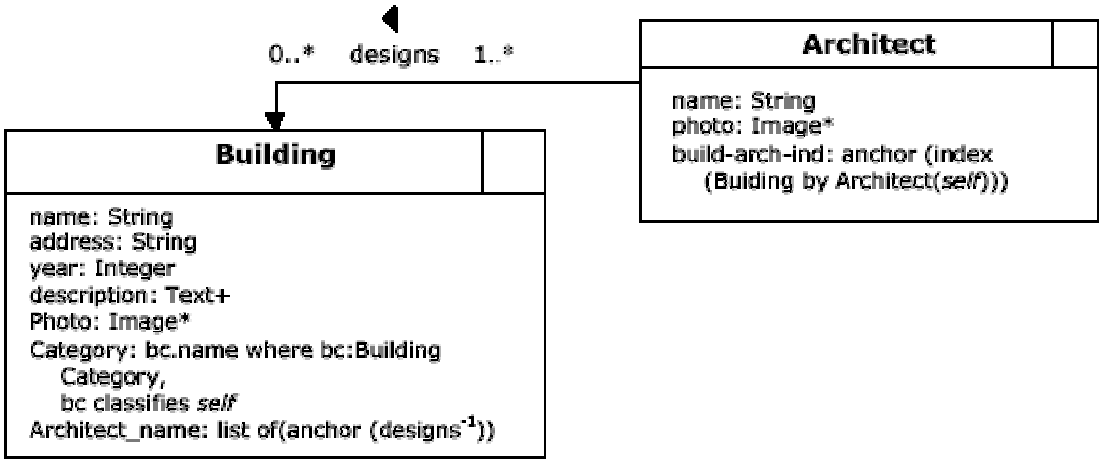

Figura 10 – Esquema de Classes Navegacionais

Fonte: Schwabe et al , OOHDM-WEB, p. 3

Analisando a figura acima, é possível observar alguns pontos interessantes:

a) Atributos multi-valorados

Sendo as classes navegacionais (nós) imagens das classes conceituais, as mesmas possuirão os atributos das classes conceituais, mas tais atributos tem que ser mapeados em separado quando são multi-valorados.

Para exemplificar, pode-se observar o atributo: description:[Text+, Photo:Image], esta é sua descrição na classe conceitual, enquanto que na classe navegacional o mesmo tem que ser separado em atributos obrigatórios e opcionais, neste caso: description:Text+ e Photo: Image * .

A afirmação acima é confirmada por Schwabe e Vilain[29] ,

> Apesar dos atributos das classes conceituais poderem apresentar múltiplas perspectivas, os atributos dos nós só podem apresentar uma perspectiva, e devem ser de um tipo predefinido simples, por exemplo, String, Integer, Real, etc... complexo, como o Array, ou definido pela aplicação. Assim, um atributo com múltiplas perspectivas de uma classe conceitual que possui uma perspectiva default é mapeado, na modelagem navegacional, para vários atributos de um nó, um para cada perspectiva. A perspectiva default é mapeada para um atributo obrigatório, semelhante aos atributos comuns. As perspectivas não default são mapeadas para atributos opcionais que são seguidos pelo símbolo " * " .

b) Âncoras e índices

As âncoras e índices de uma classe navegacional também são apresentados como atributos. Além dos elementos do contexto as âncoras e índices podem referenciar outras informações.

[29] SCHWABE , Daniel; VILAIN, Patricia. **Notação da Metodologia OOHDM.** PUC-RIO, abril 1999, P. 26.

As âncoras de acesso aos elementos de um contexto apresentam a seguinte sintaxe:

nome-âncora: âncora(índice(nome-contexto(parâmetros-contexto)))

No exemplo da arquitetura este recurso é exemplificado:

Build-arch-ind:anchor(index(Building by Architect(self)))

Segundo Schwabe e Vilain[30]: "As âncoras e índices também podem referenciar outras informações além dos elementos de um contexto"

Para exemplificar tal afirmação, o nó Building possui um atributo chamado: **arquitect_name**, o qual possui uma lista dos projetos dos arquitetos:

Arquitect_name: list of(anchor(designs^{-1})

c) Diferenças entre classes navegacionais e conceituais

Como citado anteriormente, uma das principais inovações do OOHDM é reconhecer que os objetos navegacionais são diferentes dos objetos conceituais, no exemplo da classe navegacional da arquitetura demonstra-se que o nó "Building" é uma imagem da classe conceitual "Building" e da classe conceitual "Building Category". A mesma inclui um atributo "Category" a qual consegue obter o atributo "name" da classe conceitual "Building Category", adicionando a mesma na classe navegacional "Building".

3.1.2.2 - Contextos de Navegação

Em OOHDM foi pré-definido um grupo de classes navegacionais, fazendo com que um contexto de navegação seja formado por um conjunto de objetos como: (nós, links, âncoras e estruturas de acessos). O acesso a estruturas, tais como: índices, podem representar possíveis caminhos para o usuário iniciar a navegação.

---

[30] SCHWABE , Daniel; VILAIN, Patricia. **Notação da Metodologia OOHDM.** PUC-RIO, abril 1999. P, 28.

Os contextos navegacionais apresentam-se de maneira muito importante em OOHDM, pois as ligações navegacionais poderão ser utilizadas de maneiras diferentes por usuários finais, ou seja: poderemos ter diferentes aplicações para o mesmo domínio do problema. Para exemplificar tal colocação, o website sobre arquitetura poderia ter informações sobre fórmulas complexas matemáticas, utilizadas por categorias de construção para arquitetos que visitassem o site, e não possuir tal informação para pessoas que não são da área e desejassem somente verem as obras realizadas.

Segundo Schwab et al[31]: "Quando as classes navegacionais foram finalizadas, é necessário estruturar o espaço navegacional que irá ser disponibilizado para o usuário[...]." OOHDM estrutura estes espaços navegacionais em grupos chamados de contextos, sendo que cada contexto inclui: os elementos que eles contém, a especificação de sua estrutura navegacional, um ponto de entrada, acesso e restrições de classes de usuários e suas operações.

Baseado nas informações de Schwab et al, OOHDM define seis caminhos diferentes para a exibição de contextos:

a) Derivado de classe simples

Inclui todos os objetos de uma classe que satisfaçam alguma propriedade, por exemplo: " Arquitetos com nome = Márcio " .

b) Derivado de grupo de classes

É um grupo de contextos derivados de classes simples, onde a definição da propriedade de cada contexto é parametrizada, por exemplo: "arquitetos por cidade" (a cidade pode variar).

[31] SCHWABE , Daniel; ALMEIDA, Rita; MOURA, Isabela. **OOHDM-Web: An Environment for Implementation of Hypermedia Application in the WWW**. SigWEB Newsletter, Vol. 8, #2, Junho de 1999.

c) Derivado de ligação simples

Contém todos os objetos relacionados com outro objeto, por exemplo: “Construções projetadas por Márcio Henrique Locatelli”.

d) Derivado de ligações compostas

Um grupo de contextos derivados de ligações, onde cada um é obtido variando o elemento fonte da ligação, por exemplo: “Construções projetadas por arquiteto”. Neste exemplo o arquiteto pode variar.

e) Arbitrário

Os elementos do contexto podem ser escolhidos arbitrariamente, a partir de uma ou mais classes (enumerado), por exemplo, pode-se citar um roteiro guiado dos arquitetos que possuem projetos e são adeptos da prática de esportes.

f) Dinâmico

Os elementos modificam-se conforme a navegação do usuário. Exemplo: cesta de compras, histórico.

Obs.: se existir uma estrutura de acesso para os contextos acima, a notação correspondente para representa-lo irá conter um pequeno quadro no canto superior esquerdo

Existem estruturas de acessos (índices) associadas aos contextos, as mesmas são representadas graficamente por:

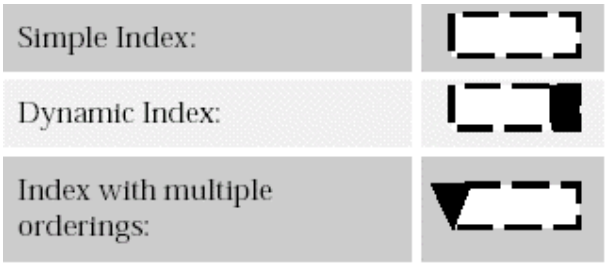

Figura 11 – Representação gráfica de índices associados aos contextos

Fonte: Schwabe et al , OOHDM-WEB, p. 4

Para exemplificar o gráfico acima, pode-se demonstrar a utilização do índice com múltiplas ordenações (Index with multiple orderings). Neste caso a estrutura de acesso poderá possuir vários critérios para sua ordenação, podendo conforme a situação alternar entre esses critérios. A figura abaixo representa um exemplo de estrutura de acesso com múltiplos critérios de ordenação.

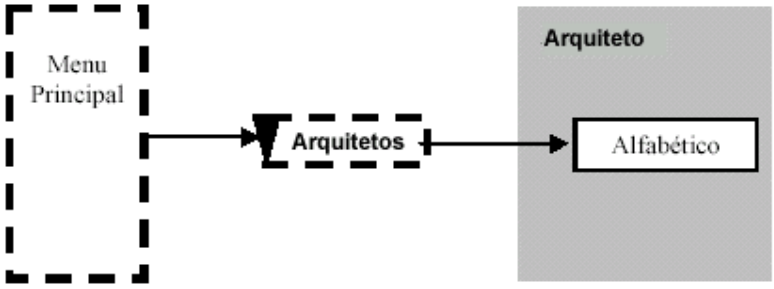

Figura 12 – Exemplo de estrutura de acesso com múltimplos critérios de ordenação

A estrutura navegacional das aplicações desenvolvidas utilizando-se OOHDM são definidas no esquema de contextos de navegação. O esquema de contextos de navegação irá demonstrar todas as estruturas de acessos, os contextos definidos para a aplicação, e também a possível navegação entre eles.

A figura abaixo demonstra o esquema de contexto de navegação para o site de arquitetura.

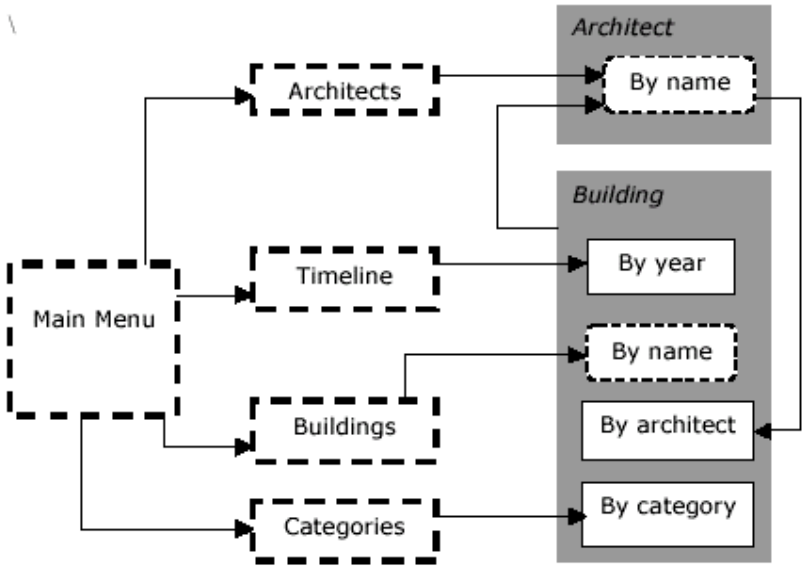

Figura 13 – Contexto de navegação

Fonte: Schwabe et al , OOHDM-WEB, p. 5

Analisando o diagrama o menu principal (Main Menu) possui quatro índices:

Architects: permite acesso a lista dos arquitetos em ordem alfabética.
Timeline: acesso aos edifícios agrupados por ano de construção.
Buildings: permite acesso aos edifícios por nome.
Categories: permite acesso aos edifícios por categoria (monumentos, hotéis,etc...)

Também é possível observar que os edifícios podem ser agrupados de acordo com o arquiteto que os projetou, este contexto pode somente ser acessado de outros contextos, como os arquitetos. Pode ser verificado também que quando olhamos um edifício em algum dos contextos, é possível escolher o contexto. Por exemplo: enquanto olhamos um edifício de um certo arquiteto é possível navegar para o próximo edifício na mesma categoria.

3.1.3 – Projeto de Interface Abstrata

No projeto de interface abstrata serão definidos os botões de controles, menus, barras de menus, etc...) enfim, todos os objetos que farão a interface com o usuário. Como o projeto de interface abstrata é executado antes da implementação deve-se ficar atento aos tipos de objetos que serão colocados na interface, pois o ambiente de implementação utilizado poderá não suportar algumas características mapeadas.

O projeto de criação de interface para o usuário final, sempre foi um dos maiores problemas na construção de WebApps, pois o alvo da elaboração dos objetos para a composição das interfaces para o usuário tem sido geralmente esboçado parcialmente e não é feito uma especificação completa dos objetos que irão compor a interface com o usuário.

Para OOHDM facilitar a modelagem de uma interface amigável com o usuário final são utilizados os ADVs (Abstract Data Views)[32], existem também objetos que são chamados de ADOs (Abstract Data objects) e não suportam eventos externos, interagindo com estruturas de dados e controle das aplicações. Existe uma associação entre os ADVs e os ADOS, conforme nos relata Schwabe e Vilain[33]: "Os ADOs estão associados a ADVs que transmitem as informações de seus estados para fora da aplicação ou para outros ADVs."

Para representar as relações entre os objetos de navegação e os objetos de interface, são utilizados os Diagramas de Configuração. Um ADV é relacionado com seu objeto correspondente na aplicação, o qual acionará um comportamento no servidor para aquelas operações não especificadas na Interface, segundo Schwab et al[34]: "Este objeto é chamado de ADV Proprietário".

3.1.4 - Implementação

Esta é a fase final da modelagem OOHDM, nesta etapa todos os modelos construídos anteriormente para serem independentes de plataforma serão agora traduzidos para um ambiente de implementação específico. OOHDM possui um ambiente de implementação específico para a web denominado OOHDMweb, o mesmo cria índices em páginas html representando os contextos a partir das informações armazenadas em tabelas.

---

[32] Objetos usados para especificar a aparência e interface dos objetos da aplicação.
[33] SCHWABE , Daniel; VILAIN, Patricia. **Notação da Metodologia OOHDM.** PUC-RIO, abril 1999, P.55.
[34] SCHWABE , Daniel; ALMEIDA, Rita; MOURA, Isabela. **OOHDM-Web: An Environment for Implementation of Hypermedia Application in the WWW**. SigWEB Newsletter, Vol. 8, #2, Junho de 1999.

3.2 –WebML (The Web Modeling Language)

WebML possui uma notação simples para a especificação de WebApps complexos a nível conceitual, ela auxilia os desenvolvedores a implementarem as principais funcionalidades de um WebSite em alto nível, sem ser necessário implementar os detalhes de mais baixo nível. Apresentando uma representação gráfica bastante intuitiva, WebML faz com que seus conceitos sejam associados facilmente a essas representações. A facilidade na manipulação das representações gráficas fazem com que várias ferramentas CASE suportem WebML, tornando assim sua utilização facilitada, pois outros membros da equipe que não conheçam profundamente de programação poderão engajar-se no projeto, como por exemplo: Designer's gráficos e produtores de conteúdo.

Segundo Ceri et al [35]: "A representação de um site em WebML consiste de quatro modelos".

- Modelo de Dados
- Modelo de Hipertexto
- Modelo de Apresentação
- Modelo de Personalização

A apresentação dos modelos supra citados baseia-se no conteúdo que encontra-se no endereço: http://webml.elet.polimi.it/webml/page1.do

3.2.1 – Modelo de Dados

WebML utiliza-se de notações clássicas para representar a modelagem de dados: modelo E/R (Entidade e Relacionamento), ODMG (Modelo orientado a Objetos), Diagrama de Classes UML.

---

[35] CERI , Estefano; FRATERNALI, Piero; BONGIO, Aldo. **Web Modeling Language: a modeling language for designing Web Sites**. Dipartimento di Elettronica e Informazione, Politecnico di Milano, 2000.

Existem dois elementos fundamentais no modelo de dados: as **Entidades** e os **Relacionamentos.**

a) Entidades

As entidades irão representar um grupo de objetos com características comuns e também podem ser organizadas em hierarquias de generalização. Em WebML uma entidade é representada conforme a figura abaixo:

Figura 14 – Representação de Entidade em WebML
Fonte: Overview of WebML – disponível em: http://webml.elet.polimi.it/webml/page1.do

O nome da entidade fica no topo, seguido pela lista de atributos. Na figura acima é representada a entidade Álbum, a qual descreve como propriedade dos objetos instanciados por essa entidade um título e uma duração.

b) Relacionamentos

Os relacionamentos representam o significado das conexões entre as entidades, como é possível observar na figura abaixo:

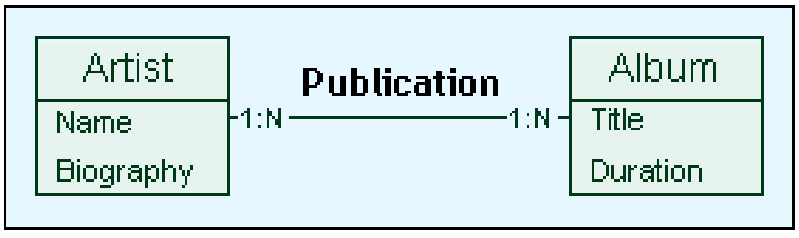

Figura 15 – Representação de Relacionamento em WebML
Fonte: Overview of WebML – disponível em: http://webml.elet.polimi.it/webml/page1.do

A figura acima representa o relacionamento entre um artista e seu álbum, o significado da associação é sabido pelo nome dado ao mesmo, neste caso utilizou-se o nome de “Publication” para demonstrar os albuns publicados por um artista.

Para demonstrar um exemplo de um modelo de dados é possível observar a figura abaixo:

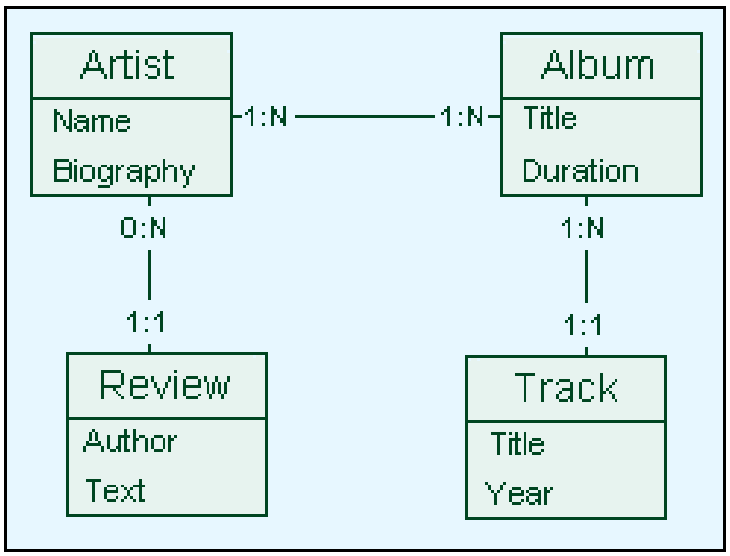

Figura 16 – Modelo de dados em WebML

Fonte: Overview of WebML – disponível em: http://webml.elet.polimi.it/webml/page1.do

A figura acima mostra a representação da informação de álbuns musicais, os quais são compostos por artistas, sobre o que são fornecidas revisões. Cada álbum poderá conter várias músicas.

3.2.2 – Modelo de Hipertexto

O modelo de hipertexto especifica a composição e a navegação do site. Neste modelo é especificado que o hipertexto é composto por páginas, que por sua vez são compostas de Units. As Units são os elementos utilizados para publicar as informações descritas no modelo de dados. As units podem ser de seis tipos diferentes: Data, multi-data, index, filter, scroller, direct-units. As Data Units publicam as informações de um único objeto, conforme demonstra a figura abaixo:

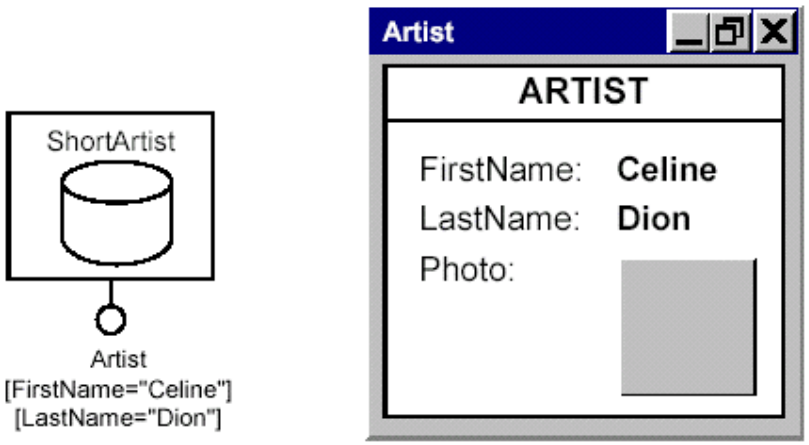

Figura 17 – Data Units

Fonte: Overview of WebML – disponível em: http://webml.elet.polimi.it/webml/page1.do

Obs.: Os outros tipos de Units representam outras maneiras de publicar e setar objetos.

A navegação também é definida no modelo de hipertexto, a navegação em um WebApps é definida através de ligações (links). Os links podem ser definidos entre as units em uma única página, entre as units estabelecidas em páginas diferentes, ou entre páginas.

3.2.3 – Modelo de Apresentação

WebML não possui um modelo específico para expressar a apresentação gráfica das páginas em nível conceitual.

Desde que as especificações de WebML podem ser representadas utilizando-se XML, a apresentação das páginas é considerada como uma transformação de documentos, constrói-se a especificação WebML de uma pagina em uma página escrita em uma linguagem de implementação concreta como: JSP ou ASP.NET. Conseqüentemente, a construção de templates (modelos) para a apresentação das páginas podem ser construídos utilizando-se XSL – Extensible Style Sheet Language[36], a utilização de XSL vem a substituir os famosos CSS – Cascading Style Sheets[37], pois o XSL adotou a sintaxe do XML, podendo assim ser incluído na especificação WebML, outro fato importante é que o XSL não manipula elementos, manipula objetos gráficos ("flow objects"). Mais especificamente transforma fragmentos de um documento XML em objetos gráficos.

3.2.4 – Modelo de Personalização

Usuários e grupos de usuários são explicitamente modelados na forma de entidades pré-definidas, chamadas de User e Group. As características dessas entidades podem ser usadas para armazenar o conteúdo de um grupo específico, ou do usuário individualmente, ver sugestões, listar favoritos e otimizar recursos gráficos.

3.2.5 – Implementando um projeto em WebML

Após serem efetuadas todas as etapas de um projeto em WebML, pode-se utilizar uma ferramenta CASE para gerar automaticamente a aplicação a partir dos diagramas construídos. Atualmente existe uma ferramenta CASE chamada *WebRatio Site Development Studio* a qual é livre para fins não comerciais e é muito intuitiva e fácil de ser utilizada.

---

[36] O XSL permite reordenar estruturalmente um documento, podendo assim fazer uma transformação do documento original.

[37] CSS é uma linguagem declarativa que permite a associar informações de estilos em documentos XML ou HTML.

4 – CONSIDERAÇÕES FINAIS

O campo de pesquisa da Engenharia de Software para o desenvolvimento de WebApps é muito amplo e desafiador, quando optou-se por tal Engenharia na realização deste trabalho, teve-se como objetivo demonstrar que as diferenças entre a Engenharia de Software Tradicional e a Engenharia para o desenvolvimento de WebApps justificam a utilização de métodos especiais para o desenvolvimento e manutenção de WebApps complexos. Muitos profissionais de Engenharia de Software ainda apresentam certa "desconfiança" para a utilização da Engenharia de Software para o desenvolvimento de WebApps, justificando que a Engenharia de Software Tradicional consegue suprir todas as etapas de um projeto de desenvolvimento de WebApps, como foi demonstrado neste trabalho tal afirmação não é justificada, pois embora a Engenharia de Software para o desenvolvimento de WebApps utilize-se de conceitos da Engenharia de Software tradicional os métodos e as técnicas tradicionais não conseguem realizar todas as etapas para a criação de projetos baseados em hiperdocumentos, outro problema é a navegação no domínio da aplicação, pois em hiperdocumentos o usuário passa a ser um participante ativo do ambiente, podendo criar seu próprio contexto navegacional.

Uma das maiores motivações na elaboração do presente trabalho foi pesquisar métodos que dariam suporte para gerenciar o projeto para a criação de WebApps, com isto foram escolhidas duas metodologias: *OOHDM – Object Oriented Hypermedia Design Method e WebMl – The Web Modeling Language*, o estudo das metodologias citadas foi de extrema importância, devido ao fato que podem servir de base para pessoas que não conhecem metodologias de desenvolvimento de WebApps poderem ver suas finalidades e sua maneira de utilização.

Não buscou-se neste trabalho a comparação entre as duas metodologias citadas, para elaborar uma comparação entre as metodologias seria necessário construir o mesmo projeto utilizando-se as duas metodologias, podendo assim justificar qual metodologia possui melhor eficácia para certas partes do projeto, com isto não quer se afirmar que uma metodologia é melhor do que a outra, pois dependendo do tipo de projeto uma metodologia poderá se ajustar melhor que a outra.

Foi possível constatar que a metodologia OOHDM e a metodologia WEBML estão em constante refinamento, sendo que vários itens que fazem parte de algumas etapas do projeto podem sofrer mudanças significativas em um curso espaço de tempo, como pode ser observado na tese de Doutorado do Sr. Gustavo Rossi(1997), se comparado com seu primeiro trabalho em 1996 sobre a metodologia OOHDM. Embora as duas metodologias apresentem formas diferentes de representar as etapas de modelagem de um projeto, as duas demonstram de maneira clara e objetiva que nunca as etapas da modelagem devem ser esquecidas ou colocadas em segundo plano, porque se isto acontecer os projetos podem ser um fracasso tanto em sua estrutura conceitual como navegacional.

De modo geral os objetivos do presente trabalho foram alcançados, pois foi possível observar as principais diferenças entre a Engenharia de Software para o desenvolvimento de WebApps e a Engenharia de software tradicional, demonstrando que o desenvolvimento de hiperdocumentos necessitam realmente de metodologias que dêem suporte a sua construção e manutenção.

4.1 - Recomendações

A utilização de ambientes específicos de implementação que suportem as metodologias apresentadas neste trabalho (exemplos disso são: OOHDMWEB para OODHM e WebRatio para Webml) podem ser interessantes para avaliar-se as principais diferenças entre as duas metodologias para a execução de um projeto específico.

Por fim pode ser recomendada para trabalhos futuros a criação de uma ferramenta CASE que seja compatível com as metodologias citadas, e trabalhe integrada com a construção e manipulação de interfaces para usuários.

REFERÊNCIAS BIBLIOGRÁFICAS

FURLAN, Davi José. **Modelagem de Objetos através da UML – Unified Modeling Language.** 1. ed. São Paulo: Makron Books, 1998. 329 p.

Natacha Güell. **Apresentação da Metodologia OOHDM**. Disponível em: <http://www.inf.puc-rio.br/~natacha/OOHDM/princ.htm>. Acesso em julho/2003.

ROSSI, Gustavo. **A Metodologia OOHDM.** Disponível em: <http://www.inf.puc-rio.br/~schwabe>. Acesso em julho/2003.

SCHWABE, Daniel. **The Object Oriented Hypermedia Design Model.** 2000. Disponível em <http://www.telemidia.puc-rio.br/oohdm/oohdm.html>. Acesso em julho/2003.

SHIRK, H.N. **Cognitive Architecture in Hypermedia Instruction**. 1. ed. EUA: Cambridge, MA: MIT Press, 1992. 112 p.

YOURDON, Edward. **Análise e Projetos Orientados a Objetos.** 1. ed. São Paulo: Makron Books, 1999. 328p.

PRESSMAN, Roger S. **Engenharia de Software.** 3. ed. São Paulo: Madron Books, 1995. 1056p.

PETERS, Peters F. Pedrycz, Witold. **Engenharia de Sofware: Teoria e Prática.** 2. ed. Rio de Janeiro, 2001. 602p.

HENNICKER, Rolf; KOCK, Nora. **A UML-based Methodology for Hypermedia Design Method**. Institute of Computer Science – Ludwig-Maximilians University of Munich, 2000.

BIZZOTTO, Eduardo Carlos. **Director 8.** 1.ed. São Paulo: Makron Books, 2000. 412p.

TURINE, Marcelo Augusto Santos. **HMBS- Um Modelo Baseado em Statecharts para a Especificação Formal de Hiperdocumentos.** 1998. 178p. Tese (Doutorado em Ciências da Computação) Universidade de São Paulo / USP.

SZWARCFITER, J.L. **Grafos e Algoritmos Computacionais.** São Paulo; Ed. Campus, 1984.

FRAÏSSÉ, Sylvain : **A Task Driven Design Method and Its Associated Tool for Automatically Generating Hypertexts**. Hypertext 1997. Disponível em: <http://www.informatik.uni-trier.de/~ley/db/conf/ht/Fraisse97> . Acesso em julho de 2003.

[Ginige 2000] GINIGE, A.; MURUGESAN, S. **Web Engineering: An Introduction.** University of Western Sydney, Australia, IEEE, 2000.

SCHWABE , Daniel; VILAIN, Patricia. **Notação da Metodologia OOHDM.** PUC-RIO, abril 1999.

GRAEF, G.; GAEDKE, G. Development and Evolution of Web-Applications using the WebComposition Process Model. International WorkShop on Web Engineering at 9th International, World Wide Web Conference (WWW9), Amsterdam , Holanda, Maio, 2000.

Nanard J., Nanard M. **Hypertext design environments and the hypertext design process**. Communication of the ACM, 1995. Vol 38 (8), (1995) 49-56.