Padronização de Interfaces Web Através dos Recursos da UML

Isolete Teresinha Dzendzik
LAC/INPE
iso@lac.com.br

José Carlos Becceneri
LAC/INPE
becce@lac.inpe.br

Mauricio Gonçalves Vieira
CCS/INPE
mauricio@ccs.inpe.br

Resumo

Este artigo apresenta um modelo de desenvolvimento de Websites, com a utilização dos recursos da UML. Em uma primeira fase o modelo trata do levantamento de requisitos feitos através dos diagramas Entidade-Relacionamento utilizados na Engenharia de Software. Na segunda fase os requisitos obtidos são transformados em arquivos e distribuídos em um modelo lógico, onde se desenvolve a estrutura de arquivos e diretórios. Este modelo é baseado nos diagramas de componentes da UML. Ainda nesta fase elabora-se a documentação e fazem-se as convenções de padronização de estrutura e de interfaces. A terceira fase é voltada ao design das interfaces. Os procedimentos apresentados neste modelo resultam em Websites com páginas mais leves e intuitivas.

1. Introdução

Este trabalho propõe um modelo de desenvolvimento de *Website*, que envolve projeto implementação e *design,* fazendo uso dos recursos da UML. Este modelo, chamado de "Modelo Web-UML" (MW-UML), envolve três fases de desenvolvimento Web, que são respectivamente a fase de levantamento de requisitos, fase de implementação e fase de design das páginas.

Na fase 1 do MW-UML são utilizados os diagramas de Entidade-Relacionamento (E-R) para entender e ilustrar o ambiente físico (empresa, produto, serviço, etc) que será explorado. A partir dos diagramas de E-R faz-se o levantamento das informações e a atribuição das mesmas à entidade a que pertencem.

Na fase 2 desenvolve-se toda a estrutura de arquivos e diretórios com base em um modelo lógico elaborado previamente com base nos diagramas de componentes utilizados na UML; faz-se a previsão dos níveis de Uniform Resourse Locator (URLs) e das portas de entrada do usuário; faz-se as convenções de *design*; define-se uma padronização para as interfaces; elabora-se a documentação com as informações pertinentes ao *Website* e as instruções de administração para os *Web designers*; faz-se também o planejamento de peso para cada página.

Na fase 3 é que aparecem os resultados do trabalho realizado nas fases 1 e 2. Nesta fase o que predomina é a arte, os princípios éticos com o usuário, a distribuição dos *links* nas páginas, os títulos, os cabeçalhos, os nomes dos *links*, as imagens, as técnicas de redação e outras opções de uso da página, como por exemplo, versões de idiomas e recursos de áudio e vídeo. A fase de *design* das páginas é chamada também de fase artística, considerando que com os mesmos requisitos obtidos nas fases 1 e 2, o resultado pode ser diferente de acordo com a criatividade e maturidade de cada profissional.

Este documento encontra-se organizado com 5 seções. Na seção 1 faz-se uma introdução aos assuntos pertinentes a este trabalho. Na seção 2 faz-se uma introdução sobre a UML. Na seção 3 encontra-se o estado da arte mostrando alguns tipos de estrutura, padrões e modelos de projetos de *Websites*. Na seção 4 encontra-se a contribuição deste trabalho através de um modelo que envolve projeto, implementação e *design* de *Websites*. Na seção 5 mostram-se algumas conclusões.

2. UML

A *Unified Modeling Language* (UML) é utilizada no mercado de *software* como uma linguagem gráfica padrão destinada à especificação, construção, visualização e documentação de sistemas de *software*. As páginas *Web* são consideradas como segmentos de *software*, onde os conceitos e as atividades práticas estão sempre recorrendo a exemplos que mostrem os princípios da interação humano-computador. Isso envolve a aplicação das principais técnicas de design de interfaces de usuário para que resultem em páginas mais leves e intuitivas.

Os recursos da UML proporcionam uma linguagem comum entre desenvolvedores e administradores de *Websites* no projeto, implementação e administração. Ao desenvolver projetos voltados a *Web*, prevendo estruturas padronizadas, obtém-se uma aproximação maior do ideal de fazer com que os *Websites* possam ser sistemas "orientados ao usuário". Isso significa sistemas que abrangem um número maior de usuários que verão um Website tal como foi desenvolvido, independente dos recursos de software, hardware e banda que possam ter [1].

3. Estruturas, padrões e modelos de *Websites*

Nesta seção explicam-se as estruturas, padrões e modelos de projetos mais utilizados em *Websites*.

3.1. Estruturas de *Websites*

Para que uma página seja elaborada é necessária a adoção de um tipo de estrutura para a mesma. A estrutura de uma página *Web*, é definida pelo esboço onde serão colocados os conteúdos [5].

Ao iniciar um projeto de *Website* é necessário definir o tipo de estrutura, pois os arquivos terão que ser desenvolvidos de acordo com as características da estrutura, como por exemplo, o uso do parâmetro "*Target*" em *links*, que define em que tipo de janela uma página será exibida.

Destacam-se a seguir, os tipos mais freqüentes de estrutura encontradas em *Websites*.

3.1.1. Página única com *links* para os tópicos

Essa é a forma mais simples de se expor conteúdos *Web*. A composição se dá com uma única página onde há *links* que apontam do início da página para um tópico específico, do final para o início e de qualquer ponto da página para onde houver um *link* (esta estrutura é bastante utilizada em *FAQs*). O limite deste tipo de estrutura encontra-se no tamanho das páginas geradas quando o conteúdo é grande e na organização dos diretórios. Quando há mais de um assunto a ser exposto e quando há necessidade de ilustrações feitas com imagens, esse tipo de estrutura não é o mais adequado.

3.1.2. *Frames*

Os *Frames* são estruturas vazias, que somente dividem a tela. Nos espaços vazios são inseridos arquivos com conteúdos. Normalmente um arquivo fica fixo, com a barra de navegação fazendo a chamada das páginas que compõem o *Website*. Este sistema é bastante utilizado devido à praticidade no desenvolvimento, porém para o usuário, apresenta problemas funcionais, éticos e estéticos.

3.1.3. Páginas divididas com tabelas

É um sistema onde se elabora um arquivo com uma barra de navegação. O *script* deste arquivo é colado em todas as páginas, fazendo com que o usuário veja as páginas escolhidas, acompanhadas da barra de navegação [2].

Este tipo de estrutura revela suas limitações em relação ao número de arquivos e diretórios que compõem um *Website*. A necessidade de copiar e colar o *script* em todas as páginas não representa grandes problemas para um *Website* com poucos arquivos, mas considerando um grande número, este sistema torna-se impraticável.

3.2. Modelos e padrões de *Websites*

Nesta seção faz-se um levantamento dos modelos e padrões de projetos de desenvolvimento de *Websites*.

3.2.1. Práticas recomendadas para a Internet (IEEE 2001)

O IEEE 2001 (Instituto de Engenharia Elétrica e Eletrônicos) é um padrão voluntário conhecido como "Práticas recomendadas para a Internet – Engenharia *Web*, gerenciamento de *Websites* e padrões de ciclos de vida de *Websites*". O padrão conhecido como IEEE 2001 incluiu os tipos de informação que devem ser destacadas em todos os *Websites*, tal como informações sobre o criador do *Website*; URLs e datas de *updates*; identificação do que é, e a quem pertence; o propósito do *Website*; se tem fins lucrativos ou é somente informativo; quem são as pessoas responsáveis pelas divisões e a forma de contatá-las; a política de privacidade, como por exemplo, se o *Website* tem *cookies* que armazenam dados do usuário; a propriedade intelectual das informações disponíveis; se as informações estarão permanentemente no *Website* ou se são temporárias e quando são temporárias, qual é a data de expiração, de atualização e a última modificação [6].

3.2.2. HDM

O *Hypertext Design Model* (HDM) (garzotto, *et all*, 1993) prescreve um esquema de aplicação que descreve as classes abrangentes de informações dos elementos quanto as características comuns apresentadas, a organização interna da estrutura e a composição das interconexões mútuas. Portanto, este esquema captura a semântica e as regularidades estruturais na representação de estruturas para uma determinada classe de aplicações.

O HDM é um dispositivo de modelagem que provê modos de descrever aplicações de hipertexto concisamente e de uma maneira independente do sistema existente. O esquema ajuda o autor a formular os conceitos de uma determinada aplicação sem uma comunicação de linguagens entre *Web designers*, desenvolvedores e usuários.

As entidades são naturalmente agrupadas em tipos de entidades, que correspondem a classes de objetos na vida real. Em aplicações de hipertexto as entidades são freqüentemente um mesmo tópico que deve ser representado em diversas formas alternativas [4].

3.2.3. OOHDM

O método *Object-Oriented Hypermidia Design Method* (OOHDM) (Rossi *et all*, 1996), é um método para desenvolver hipertextos baseado na orientação a objetos; analisa os objetos a serem colocados nos hipertextos e suas ligações; é composto por quatro atividades diferentes, sendo o projeto conceitual, projeto navegacional, projeto abstrato da interface e a implementação [8].

3.2.4. Considerações

Além dos modelos mostrados nesta seção, existem outros que são voltados à modelagem de dados em projetos para a *Web*. A dificuldade de um *Web designer* está em como transformar um projeto feito por um destes modelos de Engenharia *Web* em um produto final ou em um *Website*.

A limitação do uso de um padrão como o IEEE 2001, encontra-se na questão prática, pois somente fala do que fazer ou não fazer, mas não traz nenhum exemplo de como fazer ou em que situação usar.

4. Padronização de Interfaces *Web* através dos recursos da UML

Nesta seção, introduz-se um novo modelo de desenvolvimento de *Websites*, que é a contribuição deste trabalho. Propõe-se um modelo que especifica três níveis de atividades, mostrando a evolução de um projeto desde o levantamento de requisitos até o design final das páginas.

Este modelo resolve problemas de interfaces *Web*, como estrutura, peso, tempo de carregamento de cada página, tamanho de URLs, *design*, estética e ética procurando colocar em prática o que se chama pela Engenharia *Web* de "Sistema Orientado ao Usuário".

Diferentes dos *softwares* tradicionais, onde se estudam os recursos de *software* e *hardware* disponíveis em uma organização antes de se começar a desenvolver um produto, os *Websites* tem que ser desenvolvidos considerando "possibilidades de recursos". Por isso, buscam-se padrões de desenvolvimento que possam reduzir ao máximo a dependência dos recursos do usuário, fazendo com que uma página seja vista da maneira mais uniforme possível, independente do sistema do usuário.

Uma solução encontrada para esta necessidade é utilizar um sistema *Server Side Include* (SSI) onde a máquina servidora é responsável pela página gerada e não os recursos da máquina do usuário. Quando um usuário acessa uma página desenvolvida com SSI, a máquina cliente faz a requisição para a máquina servidora que processará a requisição, enviando para o usuário somente o HTML gerado. Assim, a resposta será a mesma para qualquer usuário, independente dos recursos que possa ter em sua máquina.

De acordo com o modelo proposto um *Website* é concluído após passar por três fases:

Levantamento de requisitos: o processo de levantamento de requisitos é feito através do modelo E-R utilizado na Engenharia de Software (ES). Como as informações podem ser provenientes de várias fontes, através de um modelo baseado em E-R é possível ver a que entidade a informação pertence e armazená-la como parte desta entidade.

Implementação: quando as informações já estiverem disponíveis e armazenadas de acordo com as entidades pertencentes, planeja-se a estrutura lógica para as informações através de diagramas de componentes utilizados na UML. A partir do modelo lógico são desenvolvidos os arquivos e diretórios a partir de um diretório *root*. Nesta fase elabora-se também a documentação do *Website*, onde se fazem as convenções de uso como valores fixos, percentuais, cores, textos, imagens, peso das páginas, critérios de ordem dos assuntos, etc... O sistema MW-UML propõe para esta fase um modelo de estrutura chamado de "Estrutura Dinâmica para Interfaces *Web*" (EDIW) que é uma estrutura de divisão com tabelas onde se incluem os arquivos que compõem uma página (Seção 4.1).

Design das interfaces: nesta fase já se tem a estrutura e a documentação, restando fazer a distribuição dos *links,* das imagens e dos conteúdos que compõem os arquivos. As duas fases anteriores são modelos feitos através dos recursos da UML e da ES e de acordo com os princípios da Engenharia Web. Nesta fase inicia-se a parte onde as decisões não são feitas com base na engenharia, porque cada situação exigirá uma decisão diferente. Esta é a fase onde a arte aparece; é a parte que será vista pelo usuário; é onde se decide como ficará a distribuição dos *links* na *Homepage* e nas demais páginas; o espaço que cada imagem ocupará na página, independente de seu peso; as frases e palavras atribuídas aos *links*, independente dos nomes dos arquivos; em que parte de um texto deve-se inserir um *link* para um outro assunto ou complemento do assunto em questão e outras questões de *design*.

4.1. Estrutura dinâmica para interfaces de *Websites*

Para que se tenha um padrão de estrutura das páginas que compõem um *Website*, o modelo MW-UML propõe o modelo de estrutura EDIW. Este modelo de estrutura é composto por arquivos-matriz que são arquivos onde não há a exposição direta de conteúdo, mas a chamada de arquivos que farão a composição das páginas. Dessa forma a barra de navegação fica sempre visível, porém sem a necessidade de que o *script* esteja diretamente em todas as páginas e sim em um único arquivo que é chamado pelos arquivos-matriz.

A composição dos arquivos-matriz é feita através do recurso de inclusão de arquivos disponíveis em *softwares* servidores de páginas dinâmicas que suportam SSI. O dinamismo se dá devido ao montante de cada página estar incluso em arquivos-matriz onde não há código exposto e sim instruções de inclusão dos arquivos de conteúdo (textos, imagens, *links*) e de interface (cores, fontes, alinhamentos, etc). Assim, as páginas são compostas dinamicamente fazendo com que toda a instrução de interface esteja em um único arquivo incluso em quantas páginas forem necessárias. No caso de mudança de interface, muda-se apenas em um

arquivo e todo o Website assume e mudança feita.

O exemplo mais tradicional de páginas *Web* é uma barra de navegação vertical à esquerda, com aproximadamente 30% do tamanho da tela, sendo o restante destinado ao conteúdo propriamente dito [5].

Para este exemplo, com o uso do EDIW, são necessários três arquivos: o arquivo com o menu ou barra de navegação, o arquivo da página principal e o arquivo-matriz que faz a chamada de ambos, resultando em uma única página, a qual é acessada pelo usuário.

Neste exemplo, os arquivos que têm instrução de *include* são mostrados com a extensão PHP, que é uma das linguagens que suportam SSI.

4.1.1. Composição dos arquivos-matriz

Default.php: o primeiro arquivo de um *Website* deve se chamar *index* ou *default*, seguido de um ponto e sua extensão, pois são os arquivos que o servidor de páginas *Web* procura automaticamente quando se aponta para um diretório [5].

A Figura 1 mostra o *script* do arquivo default.php

```
<HTML><BODY>
<TABLE BORDER=”0” WIDTH=”100%”
HEIGHT=”0”> <TR>
<TD WIDTH=”30%”> <?INCLUDE (”menu.php”);?>
</TD>
<TD WIDTH=”70%”> <?INCLUDE
(”principal.html”);?> </TD>
</TR> </TABLE></BODY> </HTML>
```

Figura 1 - Arquivo default.php.

Utiliza-se a *tag* <TABLE> para definir as dimensões de uma página. O *script* da Figura 1 resulta em duas colunas, sendo uma à esquerda, de aproximadamente 30% para a barra de navegação que está no arquivo menu.php e outra à direita com o restante da tela. O arquivo menu.php é incluso na coluna do lado esquerdo da tabela do arquivo-matriz default.php, conforme será visualizado na página.

Menu.php: quando um arquivo tiver uma instrução *include,* precisa ser salvo com a extensão PHP para que os comando sejam executados. Como o arquivo menu.php é chamado por todos os arquivos-matriz, é necessário que o mesmo tenha extensão PHP, pois nele estão inclusos os arquivos de interface e da barra de navegação. O uso de inclusão forma uma espécie de hierarquia onde o arquivo-matriz chama o arquivo menu.php, que chama os arquivos de interface que podem chamar outros arquivos para integrá-los.

Principal.html: o arquivo principal.html, bem como os demais que sejam parte integrante de um *Website*, são arquivos comuns, como se fossem desenvolvidos para qualquer outro tipo de estrutura, como as descritas na Seção 3.

4.1.2. Padronização das interfaces e URLs

Os URLs são a nomenclatura utilizada na Internet para indicar o endereço de um documento. Essa nomenclatura inclui três componentes: o protocolo, o endereço do servidor e a localização do arquivo. Por exemplo, no URL http://www.pagedemo.com.br/index.php, o http representa o protocolo, o www.pagedemo.com.br representa o servidor que será acessado e o index.php é a localização do arquivo [5].

Um dos problemas encontrados nos URLs são os tamanhos de endereços gerados devido aos diversos níveis de diretórios criados para organizar um diretório *root*. Por exemplo, um URL provindo de um terceiro nível a partir do *root*, ficaria com um tamanho de nomenclatura semelhante a este: http://www.pagedemo.com.br/empresa/cidade/setor/index.php [2].

É indiscutível a necessidade da criação de subdiretórios para se obter uma boa organização. O que o EDIW proporciona é que o usuário seja poupado da necessidade de ter que trabalhar com URLs longos, fazendo com que os endereços a serem acessados sejam somente do nível *root* e de apenas um nível abaixo onde ficam os arquivos-matriz. Estes dois diretórios fazem chamadas através do recurso de *include*, dos arquivos que estejam em qualquer outro diretório.

A Figura 2 mostra o modelo proposto, onde os diretórios de saída para o usuário encontram-se no nível *root* e em um segundo nível chamado aqui de “matrizes”.

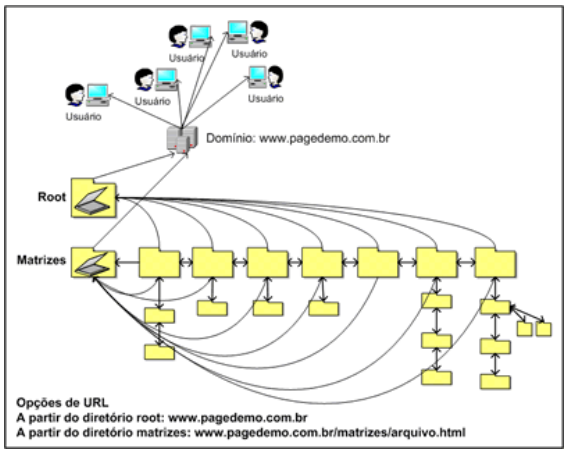

Figura 2 - Estruturas e URLs padronizadas.

Conforme mostrado na Figura 2, o usuário terá acesso somente a partir de dois diretórios, o *root* e o “matrizes” os quais contêm as instruções de estrutura e através do sistema de *include* fazem uso de arquivos de interface e de conteúdo que se encontram nos demais diretórios.

Sistemas de buscas de páginas *Web*, como o *google*, por exemplo, fazem uma varredura procurando em *meta tags* as palavras-chave digitadas pelo usuário. Muitas vezes o usuário obtém respostas

em arquivos isolados do restante do *Website* onde nem sempre há referências como data, autor, contato, atualizações, etc. Com as *meta tags* inseridas nos arquivos-matriz, os arquivos que contêm as palavras buscadas serão trazidas junto com a barra de navegação e as informações gerais do Website, pois as *meta tags* estão nos arquivos-matriz, que são compostos de interface, barra de navegação e o conteúdo específico de cada p´gina conteúdo específico de cada página.

A Figura 3 mostra o *script* de um arquivo-matriz com *meta tags* e palavras-chave no cabeçalho.

```
<HTML><BODY>
<HEAD>
<META http-equiv="Keywords" content="estrutura dinamica para interfaces de web sites">
<META name="nome site" content="conteudo da pagina">
</HEAD>
<TABLE BORDER=”0” WIDTH=”100%” HEIGHT=”0”> <TR>
<TD WIDTH=”30%”> <?INCLUDE (”menu.php”);?>
</TD>
<TD WIDTH=”70%”> <?INCLUDE (”principal.html”);?> </TD></TR> </TABLE>
</BODY> </HTML>
```

Figura 3 - Arquivo-matriz com *meta tags* no cabeçalho.

4.1.3. Recursos necessários

Para o desenvolvimento de um *Website* com o EDIW, pode-se utilizar um Sistema Operacional como o Windows NT/2000/XP que utiliza o servidor IIS que suporta ASP. Para utilizar PHP é necessária a instalação do suporte. Outra opção são os sistemas Unix/Linux com o servidor Apache que já traz a PHP por *default*.

O EDIW dispensa ou torna opcional a utilização de softwares geradores de códigos, pois os conteúdos ficam em arquivos independentes de arquivos de interface. Com isso o administrador não precisa saber mais que simples comandos de HTML, como tabelas, formatações de parágrafos e instruções de fontes. Por isso pode-se utilizar um editor de texto qualquer e um navegador como o *Internet Explorer* ou o *Netscape* para a visualização.

Para elaboração de estruturas dinâmicas pode-se utilizar simplesmente HTML e PHP, fazendo uso de um editor de texto e de um navegador para visualização. Linguagens como JavaScript, CSS, Java e demais linguagens *Web*, podem ser utilizadas na composição dos conteúdos integrantes.

Os arquivos que trazem as instruções de interface e de conteúdo podem ter extensões html, txt, xml, js, css etc. Entretanto os arquivos que têm instruções para chamar outro para que lhe seja parte integrante, precisam ter extensão PHP, pois se trata de um recurso que precisa ser interpretado pelo servidor de páginas PHP [3].

4.1.4. Utilização de estruturas dinâmicas de interfaces *Web*

O modelo EDIW pode ser utilizado para finalidades como páginas pessoais, comerciais e até em grandes portais onde há conteúdos que precisam ser atualizados com bastante freqüência.

O custo de desenvolvimento e administração com um sistema de estrutura dinâmica, está mais centrado em Recursos Humanos, por não ter a necessidade de utilização de um software específico. Assim, pela sua estética, funcionalidade, praticidade e baixo custo, pode ser utilizado tanto em *Websites* comerciais como nos que são desenvolvidos e mantidos sem fins lucrativos.

A tecnologia evolui muito rapidamente e isso significa que a tecnologia usada hoje poderá não ser a mesma usada daqui a alguns anos. Já o conteúdo que compõe um *Website* será o mesmo, assuntos novos serão acrescentados todos os dias, mas os históricos serão sempre os mesmos [7].

Com a utilização de estruturas dinâmicas e padronizadas, como a proposta pelo EDIW, os conteúdos podem ser utilizados com várias linguagens existentes, como por exemplo, JSP e ASP. Poderão ainda ser reutilizados conforme as tecnologias existentes evoluam ou outras sejam desenvolvidas.

5. Conclusão

O modelo de desenvolvimento proposto foi implementado nos seguintes *Websites*:

Utilizando a linguagem ASP: http://www.neuron.com.br: (somente o modelo de estrutura dinâmica)

Utilizando a linguagem PHP: trabalho em desenvolvimento na página do INPE, divisões CAP/LAC. http://www.lac.inpe.br e http://www.cap.inpe.br. (levantamento de requisitos, implementação e design).

Alguns resultados obtidos através da utilização do modelo MW-UML podem ser vistos em questões como princípios éticos; distinção de procedimentos como arte x engenharia, estrutura x *design*; um modelo de documentação que possa servir de guia para desenvolvedores e administradores. Maiores detalhes destes resultados encontram-se nos itens a seguir.

Princípios éticos: o modelo possibilita o planejamento de interfaces leves, intuitivas e de fácil navegação, pois dá ênfase a interface que é o que o usuário acessa, entende e utiliza. Neste modelo é possível planejar princípios éticos, no peso das páginas; nos nomes atribuídos aos *links*, títulos e cabeçalhos; na ponderação do uso das cores; na estrutura que é sem divisões de tela, sem acoplamentos de janelas e sem barras de rolagens desnecessárias.

Arte x engenharia: há autores que defendem que

os os *Websites* são produtos da engenharia e que devem ser planejados, desenvolvidos e entendidos como engenharia. Outros autores defendem o princípio da arte, dizendo que as páginas *Web* são primeiro entendidas, depois lidas. Há páginas onde a arte é o motivo, ou páginas onde a arte é que consegue proporcionar um bom entendimento da página.

Este modelo considera que os *Websites* devem ser entendidos pelo usuário como uma arte, mas desenvolvido como engenharia, mostrando nas fases do projeto onde termina a engenharia e começa a arte. A realidade da *Web* não é um ou outro, mas a combinação de ambos, cada um usado em sua fase específica, fazendo com que o usuário entenda a arte e não a engenharia.

***Design* x estrutura**: Os termos design e estrutura são usados de forma confusa pela maioria dos escritores, justamente porque a estrutura não faz parte dos projetos, como visto nos modelos mostrados na Seção 4. No MW-UML a estrutura já faz parte do projeto, sendo utilizada como um esboço onde será desenvolvido o *design*. O *design* é feito dentro de um tipo de estrutura, pois representa a "decoração" de uma página, enquanto a estrutura é a forma de construção ou o "alicerce" de uma página.

Documentação: nos *softwares* tradicionais a documentação faz parte do *software* para que os usuários possam consultar, em caso de dúvida. O mesmo não acontece com os *Websites*, por isso as interfaces devem ser intuitivas e o sistema em geral deve ser fácil de ser navegado. A necessidade de uma documentação é para uso de quem está envolvido no projeto, no desenvolvimento e na administração do *Website*.

Na fase 2 do MW-UML elabora-se a documentação do *Website* englobando as três fases de desenvolvimento, inclusive um plano de treinamento para administradores. Dessa forma, mesmo que um administrador não tenha participado da etapa de desenvolvimento poderá, através da documentação, dar seqüência ao trabalho ou fazer as atualizações necessárias, consultando a documentação como se consulta em um *software* convencional.

Redução dos níveis de URLs: o modelo EDIW proporciona ao desenvolvedor e administrador a organização da distribuição dos arquivos e diretórios, conforme seja necessário para facilitar o entendimento do sistema, independente de quantos níveis sejam necessários. Para ter esta mesma organização nos demais modelos de estrutura (Seção 3.1), o usuário precisa acompanhar todos os níveis existentes na máquina, o que resulta em URLs muito longas, difíceis de serem memorizadas e confusos para serem anotados.

O modelo EDIW consegue solucionar o problema dos URLs longos fazendo com que os usuários acessem os arquivos somente de dois níveis que é o nível root e um abaixo do root onde ficam os arquivos-matrizes (Figura 2). Todos os demais arquivos constantes nos outros diretórios e sub-diretórios têm papel de "servidores de conteúdo" para os arquivos-matriz.

Redução de trabalho: o modelo MW-UML reduz o trabalho, por utilizar um modelo dinâmico de estrutura (EDIW), previsto na fase 2. Com esse modelo de estrutura não se faz necessária a repetição de instruções de interface. Estas instruções ficam em um único arquivo que é incluso nos arquivos-matriz.

6. Referências

[1] Booch, Grady. et all. *UML, guia do usuário*. Rio de Janeiro: Campus, 2000.

[2] Baker, Adam. *How to make URLs user-friendly*. Online: <http://www.merges.net/theory/20010305.html>. 2001.

[3] Br.php.net. *Manual do PHP*. Online: <http://br.php.net/tut.php>. 2003.

[4] Garzotto, F.; Schwabe, D.; Paolini P. *HDM - A Model Based Approach to Hypermedia Application Design*", ACM Transaction on Information Systems, Vol. 11, n.1, Jan., p. 1-26, 1993

[5] Franklint, Kleitor. *ASP, Técnicas e Estratégias*. São Paulo: Érica, 2001.

[6] Isaak, Jim. *Web Site Engineering Best Practice Standards* (IEEE 2001). Online: <http://personalpages.tellink.net/~isaak/2001/index.html>. 2001.

[7] Kirda, E.; Jazayeri, M.; Kerer, C. *Experiences in Engineering Flexible Web services*. IEEE Multimedia, Jan-Mar. Online: <http://www.merges.net/theory/20010305.html>. 2001.

[8] Rossi, Gustavo & Schwabe, Daniel & Barbosa, Simone. ***OOHDM**: Object-Oriented Hypermedia Design Method*, tese de doutorado, PUC-Rio. Online:<http://www-lifia.info.unlp.edu.ar/~fer/oohdm/allindex.htm>. 1996.