UTILIZANDO UML PARA WEB: UM CASO PRÁTICO

*Sérgio Akio Tanaka
*Ademir Morgenstern Padilha

RESUMO

O trabalho apresentado neste artigo foi desenvolvido no Laboratório de Engenharia de Software da UNIFIL em cooperação com o NIEEL - Núcleo de Informática Aplicada à Educação Especial de Londrina. Este projeto contempla o desenvolvimento do website do NIEEL. O website disponibiliza diversos materiais associados à educação especial, por exemplo, divulgação de trabaljos de portadores de necessidades especiais e órgãos de apoio, download de produtos desenvolvidos pelo Núcleo em caráter demonstrativo, envio de informativos por e-mail aos usuários cadastrados, além de dispor de recursos que viabilizam a comunicação entre os usuários interessados. O trabalho fundamentou–se no projeto de pesquisa do grupo de psicologia do NIELL [9]. A modelagem do desenvolvimento do projeto se baseia nos conceitos aplicados por Jim Conallen [4], de acordo com o RUP (Rational Unified Process), Processo Racional Unificado [10]. Para a modelagem da aplicação foram utilizadas a ferramenta CASE (Computer Aided Software Engineering) Rational Rose [11] e a linguagem de programação Delphi [3]. Para a análise e projeto foi utilizada a UML (Unified Modeling Language), Linguagem Unificada de Modelagem, como padrão no desenvolvimento, permitindo uma padronização na especificação, visualização, documentação e construção de artefatos de um sistema, que possam ser utilizados nos processos ao longo do ciclo do desenvolvimento [2]. Portanto, o objetivo deste trabalho foi investigar como as técnicas de modelagem de aplicações para web e o RUP podem ser aplicados em um ciclo de vida de desenvolvimento do software, utilizando um caso prático e apresentando os diagramas que fazem esta transição.

PALAVRAS-CHAVE: UML; RUP; Modelagem para WEB.

*Docente do Curso de Tecnologia em Processamento de Dados do Centro Universitário Filadélfia – UniFil.
E-mail: tanaka@filadelfia.br
E-mail: mpadilha@adetec.org.br

ABSTRACT

The work presented in this paper was developed at the Software Engineering Laboratory of UNIFIL in cooperation with NIEEL - Nucleus of Informatics Applied to Special Education of Londrina. This project comprises the development of the NIEEL website. The website provides many resources related to special education, e.g., the spreading of papers by those with special needs and support departments and organs, the download of products developed by the nucleus with a demonstrative character, news sent by email to registered users, besides providing features that make it possible for the communication among interested users. This paper is based on a research project by the NIEEL Psychology group [9]. The modeling of the project development is based on the concepts applied by Jim Conallen [4] in accordance with the RUP - Rational Unified Process [10].

The CASE (Computer Aided Engineering Software) Rational Rose tool [11], and the Delphi programming language [3] were used for the application modeling. For the analysis and project, the UML (*Unified Modeling Language*) was used as development pattern, allowing standardization in the specification, visualization, documentation, and construction of artifacts of a system that can be used in all processes throughout the development cycle [2]. Therefore, the objective of this paper was to investigate how the modeling techniques of web applications and the RUP can be applied in a life cycle of software development using a practical case and presenting the diagrams that make this transition.

KEY-WORDS: UML; RUP; WEB Modeling.

1. INTRODUÇÃO

Os objetivos da engenharia de software estão centrados na melhoria da qualidade dos processos de desenvolvimento e na qualidade dos produtos de software desenvolvidos. Outra meta importante a ser considerada é a redução dos esforços e custos envolvidos na produção de software. Neste trabalho, observou-se que é possível reutilizarmos modelos, uma vez que estes sejam bem especificados, testados e documentados.

Os recursos computacionais oferecem excelentes recursos no que se refere a atualização cultural e a aquisição de informações, que são veículos necessários para integrar o ser humano à sociedade do nosso tempo. Os recursos da Multimídia e da Internet enquadram-se neste contexto. O estímulo à educação especial através da utilização de novos recursos tecnológicos é de suma importância, já que através deles é possível orientar o indivíduo na busca constante de novos conhecimentos.

Este projeto contempla o desenvolvimento do *website* do NIEEL - Núcleo de Informática Aplicada à Educação Especial de Londrina, empregando a linguagem UML, que, através de seus estereótipos padrões, permite estender sua semântica para modelar aplicações para *web*.

Este trabalho apresenta um modelo para *web* utilizando um caso prático. A seção 2 apresenta os conceitos do RUP que foram utilizados como um guia no desenvolvimento da modelagem. As seções 3 e 4 discutem as extensões da UML para aplicações *web* e o projeto e a prototipação do *website* do NIEEL, respectivamente. A seção 5 apresenta o desenvolvimento do projeto NIEEL. Finalmente, na seção 6, as conclusões e os trabalhos futuros são apresentados.

2. RUP - *Rational Unified Process*

O RUP é um processo que fornece uma abordagem disciplinada para o desenvolvimento de software, nomeando tarefas e responsabilidades dentro de uma organização. Sua meta é assegurar a produção de software da mais alta qualidade, que satisfaça as necessidades de seus usuários finais dentro de um cronograma e orçamento previsíveis, através da integração das fases do desenvolvimento de software. As atividades do RUP dão ênfase à criação e manutenção de modelos, visando minimizar a sobrecarga associada a geração e manutenção de documentos e maximizar o conteúdo das informações relevantes [2, 12, 7, 1].

O RUP utiliza a UML para desenvolvimento dos diagramas do sistema. É nestes diagramas e suas representações que os analistas e projetistas se basearão para finalizar cada ciclo eficazmente, considerando também as mudanças que podem ocorrer após a entrega de um software em relação ao ambiente, aos sistemas operacionais, ao banco de dados e ao hardware.

A Figura 1 ilustra a arquitetura global do RUP, que representa o ciclo de vida de desenvolvimento de um software. Em cada fase ocorrem várias iterações. Uma iteração representa um conjunto distinto de atividades com um plano de linha de base e um critério de avaliação que resulta em uma versão do sistema. As fases de concepção e de elaboração abrangem as atividades de engenharia do ciclo de vida do desenvolvimento; e as fases de construção e de transição constituem sua produção. A seguir, são apresentadas as definições das fases do RUP:

concepção: estabelece o contexto de negócios para o projeto;
elaboração: estabelece um plano de projeto e uma arquitetura sólida;
construção: desenvolve o sistema;
transição: fornece o sistema a seus usuários finais.

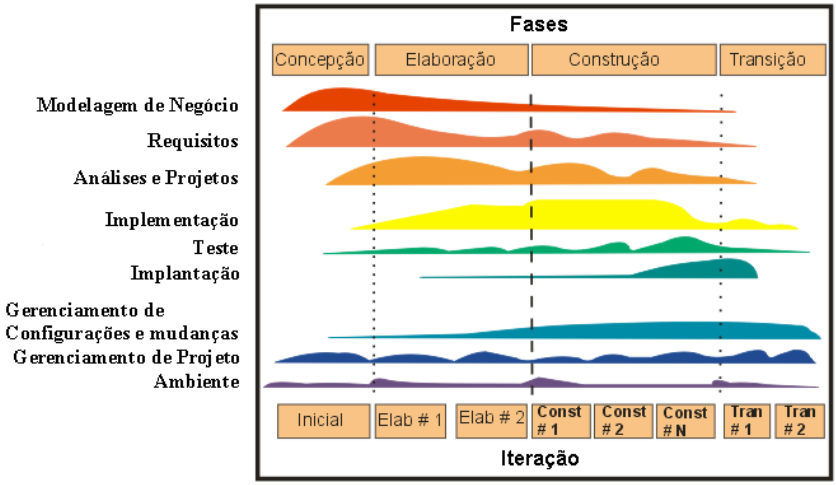

Figura 1 – O ciclo de vida de desenvolvimento de um software segundo o RUP [12].

2.1. Workflows

Os *workflows* são os tipos de artefatos mais importantes do RUP. Um *workflow* é uma simplificação da realidade, proporcionando uma melhor compreensão do sistema que está sendo criado. No RUP, existem nove modelos que, em conjunto, abrangem decisões importantes para a visualização, a especificação, a construção e a documentação de um sistema complexo de software [12]; os *workflows* são:

- **Modelo de negócio**: estabelece uma abstração da organização. É uma técnica para entender os processos de negócios de uma organização. O objetivo é encontrar os casos de uso e as entidades de negócios do software que são relevantes e que devem ser suportados pelo sistema.
- **Requisitos:** são definidos os requisitos, funcionais e não funcionais, de software de um sistema.
- **Análise e projeto**: responsável pela transformação dos requisitos em um projeto do sistema. Desenvolve uma arquitetura robusta para o sistema e adapta o projeto ao ambiente de implementação.

- **Implementação**: o modelo de implementação é uma coleção de componentes e subsistemas. Um componente representa uma parte física de implementação de um sistema, inclusive código de software (fonte ou executável) ou equivalentes, como manuscritos ou arquivos de comando.
- **Teste**: estabelece os caminhos pelos quais o sistema é validado e verificado.
- **Implantação**: abrange a configuração do ambiente e do sistema a ser entregue;
- **Gerenciamento de configuração e mudanças**: é referenciada por fornecer a linha base para a construção do produto e os seus mecanismos para controlar pedidos de mudanças, que são gerados como resultados de beta-testes e testes de aceitação.
- **Gerenciamento de projetos**: definição do escopo do projeto e acompanhamento em todas as fases do ciclo de vida do desenvolvimento do sistema.
- **Ambiente**: fornece o ambiente para que seja realizado todo o ciclo de vida do desenvolvimento do sistema.

Para estabelecer um padrão na comunicação e facilitar o entendimento do projeto, foi elaborado um *glossário*, artefato compreendido na fase de concepção do modelo de negócios, onde são definidos os termos relevantes utilizados no desenvolvimento do projeto.

A *lista de características*, artefato elaborado na fase de concepção do modelo de negócios, contempla um conjunto de atividades que permitem identificar as necessidades do usuário de modo a obter uma definição clara das características (requisitos) do sistema. Essas características descrevem o sistema em termos de funcionalidades, desempenho esperado, restrições de projeto, níveis de qualidade esperados e interfaces com outros elementos do sistema.

O artefato *requisitos adicionais*, realizado na fase de concepção do modelo de negócios, apresenta as descrições dos requisitos necessários para atender o propósito do sistema em termos de ambiente, dinamismo, prazo de entrega e segurança.

O *modelo de negócios*, artefato concebido na fase de concepção do modelo de negócios, documenta os processos de negócios do NIEEL. O objetivo é encontrar os casos de uso e as entidades de negócios do software que são relevantes e que devem ser suportadas pelo sistema.

Nas fases de concepção e elaboração, foram realizados dois modelos de casos de uso, contemplando a divisão do sistema em dois subsistemas: o controle de dados e o *website* do NIEEL. O controle de dados tem a finalidade de administrar o banco de dados do sistema independente da internet. Já o *website* é a

parte visível do sistema na *web*. Esta subdivisão é propagada para os diagramas de atividades, diagramas de interação (seqüência e colaboração), diagramas de estados, diagramas de classes, projeto de banco de dados e projeto de *interface* gráfica.

O projeto *website* do NIEEL foi desenvolvido utilizando o ciclo de vida do RUP. O *workflow* da análise e do projeto foi o foco principal do trabalho e onde foi feita a modelagem do sistema para *web*; assim, o estudo desta seção foi necessário para se poder visualizar a modelagem para a *web* dentro do ciclo de vida do processo utilizado, além da clareza e qualidade do desenvolvimento. A seção seguinte apresenta os principais conceitos da modelagem para *web* que foram utilizados no projeto proposto.

3. MODELAGEM DE APLICAÇÕES WEB COM UML

A UML é uma linguagem para modelagem de sistemas de software intensivos. Para a modelagem de páginas *web*, algumas extensões foram implementadas, utilizando estereótipos de acordo com o modelo empregado.

Os criadores da UML reconheceram que o padrão UML não se ajustava perfeitamente a todos os tipos de aplicações, e, assim, visando satisfazer as necessidades destas situações especiais, definiram um modo formal para estender tais funcionalidades através de uma semântica diferente aplicada aos elementos de modelagem. As aplicações *web* representam uma destas situações. Neste trabalho, foi tomada como base a extensão que Jim Conalen [4] definiu para o desenvolvimento de aplicações *web*.

3.1. Extensões da UML

A UML definiu um mecanismo para permitir que certos domínios possam estender a semântica de elementos a modelos específicos. O mecanismo de extensão permite a inclusão de novos atributos, de diferentes semânticas e de restrições adicionais. Parte do mecanismo de extensão de UML é a habilidade para nomear ícones diferentes a classes estereotipadas.

A extensão da UML para *web* define um conjunto de estereótipos, valores etiquetados (*tagged values*) e regras que permitem a modelagem de aplicações *web*. São aplicados estereótipos e regras a elementos que são particulares a aplicações *web*. Isto permite que estes elementos sejam representados nos mesmos modelos e diagramas que descrevem o restante do sistema.

Estereótipos podem especificar ícones especiais para serem utilizados em diagramas, ajudam na identificação de "elementos especiais" em modelos de diagramas, ou seja, é um adorno que nos permite definir um significado semânti-

co novo para um elemento do modelo.

Os valores etiquetados são grupos de valores fundamentais que podem ser associados a um elemento modelado, permitem "etiquetar" (*tag)* algum valor sobre um elemento do modelo. Um valor etiquetado está representado em um diagrama como uma *string* entre os sinais de maior e menor (ex.<*string*>).

Restrições são regras que devem ser seguidas ao agrupar os elementos do modelo. Elas podem ser expressas como texto livre entre chaves (<string>).

3.2. Arquitetura para Aplicações *web*

Segundo [4], alguns padrões foram definidos, testados e reutilizados de acordo com a arquitetura utilizada. Três padrões foram definidos especificamente para a arquiteturas *web: Thin Web Client, Thick Web Client e Web Delivery*. O padrão *Thin Web Client* é utilizado para aplicações padronizadas nas quais os objetos estarão sendo executados no servidor *web*. Já o padrão *Thick Web Client* permite que alguns objetos sejam executados no cliente. Finalmente, o padrão *Web Delivery* é utilizado para sistemas de objetos distribuídos. Neste projeto foi utilizado o padrão *Thin Web Client,* por ser mais indicado para aplicações para internet.

Na visão do cliente, a página *web* é um documento HTML (*Hyper Text Markup Language*) formatado. No servidor, porém, uma "página" pode manifestar-se de vários modos diferentes.

A arquitetura básica de uma aplicação *web* inclui: *browsers*, uma rede e um servidor de rede. *Browsers* solicitam "páginas" ao servidor. Cada página é uma mistura de conteúdo formatado de instruções expressas em HTML. Algumas páginas clientes incluem *scripts* que são interpretados pelo *browser*. Estes *scripts* definem o comportamento dinâmico adicional para a página em exibição e freqüentemente interagem com o *browser*. O usuário interage com o conteúdo da página. Às vezes o usuário insere a informação em elementos de campo na página e os submete ao servidor para processamento. O usuário também pode interagir com o sistema, navegando entre páginas diferentes através de *hyperlinks*. O usuário é a contribuição abastecedora ao sistema que pode alterar o "estado" do mesmo.

3.3. Modelando Páginas *web*

Modelos nos auxiliam a entender o sistema, simplificando alguns dos detalhes. A escolha do que modelar tem um efeito significativo na compreensão do problema e na busca de sua solução. Aplicações *web* são representadas, assim como outros sistemas, com um conjunto de modelos.

A modelagem deve fornecer subsídios que tragam benefícios aos usuários do modelo. O modelo do interior do servidor de rede ou dos detalhes do *browser* não ajudará os desenhistas e arquitetos de uma aplicação *web*. Porém, a modelagem das páginas do cliente com suas ligações e seu conteúdo dinâmico é importante. São estes os artefatos projetados pelos desenhistas, que são utilizados como instrumento de implementação. Páginas, *hyperlinks* e conteúdo dinâmico no cliente, e no servidor, é o que deveria ser modelado.

Páginas *web*, *scripts* ou páginas compiladas são elementos em UML. Um elemento é a "parte física" e substituível do sistema. A visão de implementação (Visão do Elemento) do modelo descreve os elementos do sistema e seus relacionamentos. Em uma aplicação *web*, esta visão descreve toda a rede, o sistema e as suas relações (*hyperlinks*).

Os elementos representam o empacotamento físico das *interfaces*, eles não são satisfatórios para modelar as colaborações dentro das páginas. Cada página *web* é uma classe da UML na visão de projeto do modelo (visão lógica), e suas relações para outras páginas (associações) representam *hyperlinks*. Esta abstração considera que qualquer página *web* pode representar um conjunto de funções e colaborações que só existem no servidor, e um conjunto completamente diferente que só existe no cliente.

O comportamento lógico de uma página *web* no servidor é completamente diferente da página do cliente. Enquanto executado no servidor, tem-se acesso aos recursos de páginas disponíveis no servidor (por exemplo, bancos de dados e sistema de arquivos). Em uma página *web* (ou a produção de HTML daquela página), o cliente usa um comportamento e um conjunto de relações completamente diferente. No cliente, uma página de *script* tem relações com o *browser* pelo DOM - Modelo de Objeto de Documento - e com Java Applet, ou controle ActiveX que a página tenha especificado.

Pode-se modelar o servidor e o cliente como páginas *web* através de classes, usando o mecanismo de extensão da UML para definir estereótipos e ícones para cada um, isto é, «server page» e «client page». Classes estereotipadas e ícones podem ser utilizados em um diagrama da UML, ou o nome do estereótipo, representado por *guillmets* («»), pode ser adicionado no diagrama. Os ícones são úteis para avaliar os diagramas quando atributos e operações de classes são representados neles.

Para páginas *web*, os estereótipos indicam que as classes são uma abstração do comportamento lógico de uma página no cliente ou no servidor. As duas abstrações são relacionadas entre si através de uma relação direcional. Esta associação estereotipada é denominada «*builds*» desde que uma página do cliente seja construída por uma página do servidor. Cada página *web* dinâmica é

construída com uma página de servidor. Toda página de cliente é construída por uma única página de servidor, porém, é possível uma página de servidor construir páginas de cliente múltiplas.

Um *hyperlink* em uma aplicação *web* é representado por uma associação estereotipada «link». Esta associação origina-se de uma página de cliente e aponta para uma página de cliente ou uma página de servidor.

Valores etiquetados são usados para definir os parâmetros que são passados junto com um pedido de ligação. O «link» *tagged value* "Parameters" é uma lista de nomes de parâmetros (e valores opcionais) esperados e usados pela página do servidor que processa o pedido.

Usando estes estereótipos, torna-se mais fácil modelar páginas de *script* e seus relacionamentos. Com o «server page», as operações de classe se tornam funções no servidor da página *script*, e seus atributos tornam-se variáveis de escopo de página (globalmente acessíveis pelas funções da página). Com o «client page», as operações de classe e atributos tornam-se funções e variáveis igualmente visíveis no cliente. A vantagem fundamental de separar os aspectos do servidor e do cliente de uma página em classes diferentes está na relação entre páginas e outras classes do sistema. Páginas de cliente são modeladas com relações para recursos de cliente (i.e., DOM, Java Applets, controle ActiveX). Páginas de servidor são modeladas com relações para servidor de recursos, elementos de acesso de banco de dados e sistema operacional de servidor.

3.4. Formulários

O mecanismo de entrada de dados principal para páginas de *web* é o formulário. Formulários estão definidos em um documento de HTML com *tags* <form>. Cada formulário especifica a página que é submetida para o servidor. Um formulário contém vários elementos de entrada os quais são expressos com *tags* HTML. As *tags* mais comuns são o <input>, o <select> e o <textarea>. A *tag* de entrada é sobrecarregada podendo ser: *text field, checkbox, radio button, push button, image, hidden field*, como também alguns outros tipos menos comuns. Para a modelagem do formulário, utiliza-se outro estereótipo de classe: «Form». Um «Form» é utilizado em qualquer operação na qual poderia ser definida uma *tag* <form> existente na página de cliente. Os elementos de entrada de um formulário são todos os atributos estereotipados da classe «Form». Um «Form» pode ter relações com *Applets* ou controles de ActiveX que agem como controles de entrada. Cada formulário também tem uma relação com uma página de servidor, ou seja, a página que processa a submissão do formulário. Nesta relação é utilizado o estereótipo «submit». Desde que formulários são completamen-

te contidos em um documento HTML, eles são expressos em um diagrama da UML como uma forma de agregação.

3.5. Frames

Frames permitem que múltiplas páginas sejam ativas e visíveis ao usuário em um determinado momento. Usando *scripts* do tipo *Dynamic* HTML (DHTML), os elementos nestas páginas podem interagir entre si. O potencial para interações complexas no cliente é significativamente maior e a necessidade de modelar isto também.

A utilização de *frames* como exemplos de *browser* múltiplos em uma aplicação *web* é uma decisão para o engenheiro de software. O modelo deste comportamento do cliente precisa ser representado no modelo de análise e projeto. Para modelar o uso de *frames*, mais dois estereótipos de classe foram criados, «*frameset*» e «*target*», e um estereótipo de associação, «targeted link». Uma classe de *frameset* representa um objeto de recipiente e direciona diretamente ao HTML a *tag* <*frameset*>. A classe *frameset* contém páginas de cliente e *targets*. Uma classe *target* é um *frame* nomeado ou exemplos de *browsers* que são referenciados através de outras páginas de cliente. Uma associação de *targeted link* é um *hyperlink* a outra página, mas que é feito com um objetivo específico.

4. ANÁLISE E PROJETO DO DOMÍNIO WEBSITE DO NIEEL

A modelagem de aplicações para *web* apresenta uma complexidade peculiar. Esta complexidade precisa ser compatibilizada com outros modelos da UML de forma a proporcionar uma padronização ao longo de todo o ciclo de desenvolvimento de software.

Para atingir seus objetivos, o *website* disponibilizará textos dissertativos, publicação de materiais técnicos, trabalhos de pesquisas específicos, *links* sobre eventos, assuntos ligados à educação especial, divulgação de trabalhos de portadores de necessidades especiais e órgãos de apoio, produtos de informática direcionados à educação especial, *download* de produtos desenvolvidos pelo Núcleo em caráter demonstrativo e encaminhará informativos por e-mail aos usuários cadastrados, além de dispor de recursos que viabilizem a comunicação entre os usuários interessados.

O diagrama de classes desenvolvido no modelo de análise é considerado a

contribuição primária às atividades de projeto subseqüentes.

A Figura 2 apresenta um diagrama de classe de análise, que compõe o modelo de análise desenvolvido durante a fase de elaboração. Este diagrama apresenta a visão do projetista em relação à funcionalidade que permite consultar produtos do NIEEL, denotando a interação do ator (colaborador do NIEEL) com o requisito do sistema representado pelo caso de uso "ConsultarProdutos". A rastreabilidade é demonstrada pela aplicação do relacionamento de dependência estereotipada <<trace>>, entre o caso de uso e a realização "rConsultarProdutosAnálise". Esta realização documenta a passagem da visão do usuário, documentada pelo analista de sistemas no caso de uso, para a visão do projetista, documentada nas classes. A classe de fronteira "fConsultarProdutos", que utiliza o estereótipo <<boundary>>, representa a interface com o usuário. Finalmente, a classe persistente "pProdutos" com o estereótipo <<entity>> representa a persistência das informações utilizadas pelo sistema.

A Figura 3 apresenta o diagrama de classe, elemento do modelo de projeto desenvolvido na fase de elaboração. A realização "rConsultarProdutoAnálise" é proveniente do diagrama de classe da análise. Toda realização do modelo de análise deve ter sua realização correspondente no modelo de projeto. A realização "rConsultarProdutoProjeto" da Figura 3 está ligada ao pacote ConsultarProdutos, demonstrando a rastreabilidade deste requisito pela dependência estereotipada <<trace>>.

Na Figura 4 é apresentado um diagrama de classes de projeto para aplicações *web*. Na ilustração, o uso de *frames* é empregado como exemplos de *browser* múltiplos em uma aplicação *web*, além das classes estereotipadas com «frameset» e «target» e o estereótipo de associação «targeted link».

No exemplo, tem-se uma visão em um *browser* que emprega dois *frames*. Um *frame* é nomeado com um *target* (menu), contendo uma página de cliente. O outro *frame* é nomeado com um *target* (conteúdo). Este *frame* (menu) da página de cliente é o índice do *site*. São direcionados *hyperlinks* nesta página, de forma que eles demonstrem o *frame* (conteúdo), contendo o que a será exibido na página de acordo com a opção selecionada no menu. O efeito é um índice estático no lado esquerdo e o conteúdo das páginas apresentadas dinamicamente à direita da página.

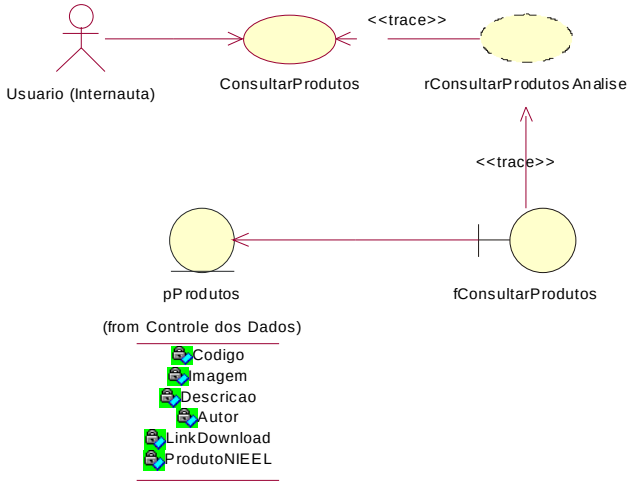

Figura 2 – Diagrama de Classe de Análise do Caso de Uso Consultar Produtos.

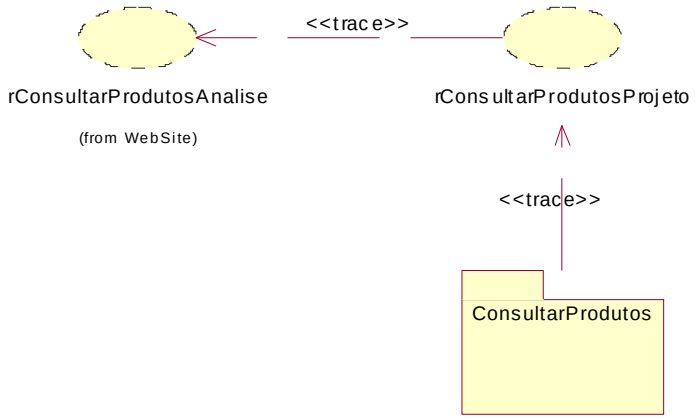

Figura 3 – Diagrama de Classe de Projeto do Caso de Uso Consultar Produtos.

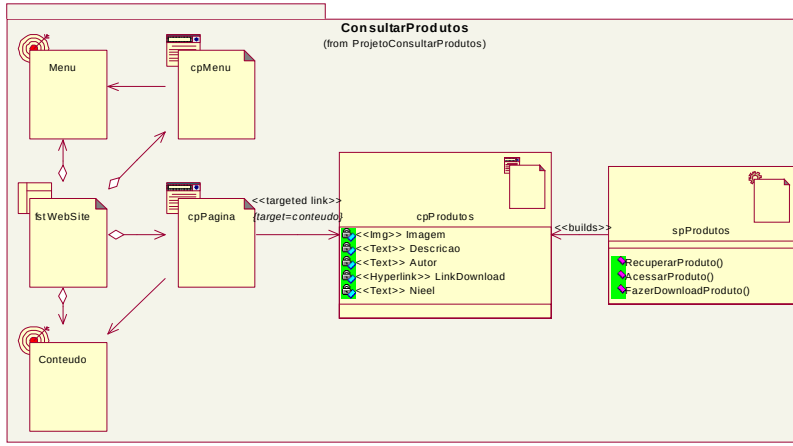

Figura 4 –Diagrama de Classe Estereotipada para Aplicações Web Consultar Produtos.

5. DESENVOLVIMENTO DO PROJETO NIEEL

Para diagramação da análise e projeto do *website* do NIEEL foi utilizada a ferramenta CASE Rational Rose 2001 [11]. As interfaces gráficas foram desenvolvidas na linguagem de programação Delphi 5.0 [3] e banco de dados ACCESS 2000, com acesso via interface JDBC - *Java Database Connectivity* e ODBC - *Open Database Connectivity* [8].

A Figura 5 mostra uma das interfaces gráficas do *web site* do NIEEL que apresentará os produtos desenvolvidos pelo NIEEL, por alunos especiais, e em outras instituições com interesses afins. O menu à esquerda da interface mostra as opções disponíveis no *web site.*

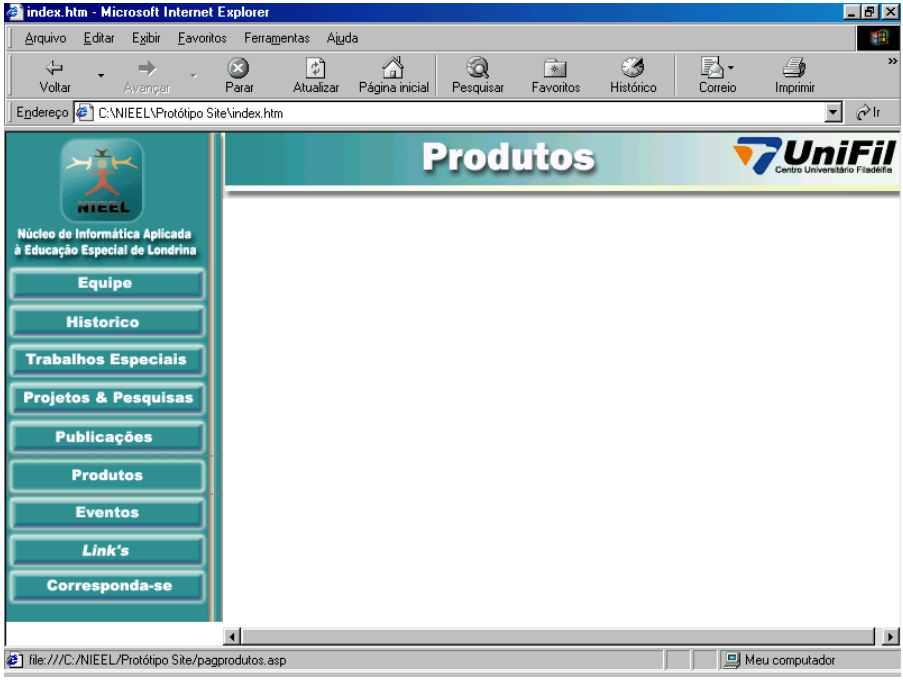

Figura 5 – Interface Gráfica de produtos do *web site* do NIEEL.

6. CONCLUSÕES E TRABALHOS FUTUROS

A elaboração do projeto, adotando-se os modelos do RUP e a documentação sistemática e disciplinada da UML, durante o ciclo de desenvolvimento do sistema, permitiu identificar, através dos documentos de entrada e saída de cada fase, a interação entre os elementos do projeto em suas diversas etapas de desenvolvimento, de forma clara e precisa, uma vez que o uso de ferramentas isoladas poderia oferecer apenas visões e soluções parciais, ou seja, o RUP atende a captura dos requisitos e a rastreabilidade do processo, mesmo em aplicações voltadas a *web*.

A adoção de um processo baseado em modelos e padrões conhecidos e aceitos reduz a prática de habilidades individuais de caráter empírico, homogeneizando a comunicação entre os diversos integrantes do processo, racionalizando técnicas de documentação e maximizando o conteúdo das informações relevantes.

As classes de fronteira são uma excelente abstração para o analista de sistemas sobre classes de projeto voltadas para web que exigem um conhecimento específico do projetista e do arquiteto, o qual envolve detalhes da linguagem (HTML) e do mecanismo do funcionamento do *browser*. Tais detalhes, se exigidos do analista de sistemas, poderiam influenciar a análise e os requisitos, com considerações de implementação.

Por outro lado, o uso das extensões da UML para modelagem de aplicações *web* viabilizou o projeto da arquitetura em um nível de detalhe e abstração apropriados aos projetistas de aplicações *web*, representando este tipo de domínio de maneira consistente.

À medida que os sistemas de software crescem em complexidade, principalmente em aplicações *web*, o processo de desenvolvimento também se torna mais complexo e passa a ser imprescindível a utilização de ferramentas e técnicas para apoiar a realização das tarefas de desenvolvimento, de modo a se obterem a qualidade e a produtividade desejadas.

O RUP, por se tratar de um processo flexível, adaptou-se perfeitamente ao contexto, integrando as necessidades de especificações do desenvolvimento para ambiente *web*, com o restante da aplicação. Isto confirma sua aplicabilidade mesmo em projetos com ambientes heterogêneos.

O uso do RUP juntamente com as extensões da UML para WEB permitiu adequar as visões do sistema de acordo com os requisitos dos papéis desempenhados pelos diferentes atores do processo. A visão do engenheiro de requisitos (modelo de caso de uso) é complementada pela visão do projetista (modelo de projeto). Esta separação de papéis é importante porque são funções e habilidades distintas que, se não forem bem coordenadas, podem interferir na construção do sistema, por exemplo, colocando restrições de projeto já na fase de concepção.

As colaborações do modelo de análise e projeto tornam a transição entre os modelos mais objetivos, sendo que uma conseqüência disto é a rastreabilidade obtida durante todo o processo de software.

Padrões e *frameworks* podem ser construídos e/ou utilizados em desenvolvimentos de aplicações para *web*. Um exemplo disso foi o padrão usado neste projeto, o *Thin Web Client*, que, uma vez aplicado, observou-se que podemos ter um ganho significativo na produtividade. Além disso, o uso de ferramentas CASE é fundamental para o desenvolvimento destes tipos de aplicativos, pois, permite a geração de código automático e reengenharia dos sistemas, aumentando, assim, a produtividade com um baixo custo.

Como trabalhos futuros, pretende-se explorar a aplicação de outros padrões de processos de software, como, por exemplo: Catalysis [5], Object-Oriented Software Process (OOSP) e Wisdom [6].

7. REFERÊNCIAS BIBLIOGRÁFICAS

AMBLER, S. W. http://www.ambysoft.com. Último acesso em 18/08/2001.

BOOCH, G., RUMBAUGH, J.; JACOBSON, I. **Unified modeling language user guide**. Addison Wesley, 1999.

CANTÚ, Marco. **Dominando o Delphi 5 – a Bíblia**. São Paulo: Makron Books,2000.

CONALLEN, J. **Building Web applications with UML**. USA: Addison Wesley, 2000.

D'SOUZA, D.; WILLS, A. **Objects, components and frameworks with UML - the catalysis approach**. USA: Addison-Wesley, 1999.

HARMELEN, M. V. **Object modeling and user interface design**. Editora Addison-Wesley, Março de 2001.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **The unified software development process**. USA: Addison-Wesley, 1999.

JSP Brasil. www.jsp.com.br. Último acesso em 15/09/2001.

JULIANI, J. **Efeitos da modalidade sensorial do estímulo nodal e da exposição sucessiva a arranjos de treino na formação de classes de estímulos equivalentes**. 1999. Tese de doutorado apresentada ao Instituto de Psicologia da Universidade de São Paulo.

KRUCHTEN, P. **The rational unified process – an introduction**. Editora Addison-Wesley, 1999.

RATIONAL Software Corporation. **UML notation guide**. Disponível em http://www.rational.com/uml/html/notation. Último acesso em 24/02/2001.

RATIONAL Software Corporation. **Rational unified process**. **Version 2001.03.00**. Copyright © 1987 – 2000. Portions © Copyright IBM Corporation 1999-2000.