Área Temática : **Inovação e Gestão Tecnológica**

Modelagem de *Web-Based Systems* (WBS): Exemplos dos Principais Diagramas UML para um Modelo do Fluxo de Informações em uma Revista Eletrônica

AUTORES
SILVIO CARVALHO NETO
Centro Universitário Uni-Facef
silvio@facef.br

ANTONIO GERALDO DA ROCHA VIDAL
Universidade de São Paulo
vidal@usp.br

HIROO TAKAOKA
Universidade de São Paulo
takaoka@usp.br

Resumo

O artigo tem como objetivo principal apresentar exemplos dos principais diagramas estáticos e dinâmicos usados na linguagem UML - *Unified Modeling Language* de modelagem conceitual e física de sistemas *Web* (*Web*-Based Systems - WBS) com programação Orientada a Objetos (OO). Para tanto, por meio de pesquisa qualitativa com dados secundários, apresenta uma revisão da teoria que aborda a descrição dos sistemas *Web* com tecnologia cliente/servidor, expõe as tecnologias disponíveis para desenvolvimento de sistemas *Web* e apresenta os conceitos e os principais diagramas das técnicas de modelagem UML. Por fim, é feito um estudo de caso de uma revista eletrônica já disponível online, com apresentação dos principais problemas encontrados no desenvolvimento desta revista eletrônica e de exemplos de diagramas que auxiliariam o desenvolvimento e a adequação desta revista para um sistema dinâmico disponível online com base no desenvolvimento de sistemas Orientados a Objetos. Os resultados da pesquisa mostram que os diagramas de UML apresentados em relação à revista eletrônica consistem em um primeiro passo que facilitat o desenvolvimento de funcionalidades diversas da revista eletrônica para os diversos atores envolvidos no sistema.

Palavras-Chave
Sistemas *Web* – Modelagem UML – Revista Eletrônica

Abstract

The article has as central objective to present main examples of static and dynamic diagrams from UML - Unified Modeling Language, language for conceptual and physical WBS modeling (Web Based Systems Modeling) especially with Oriented Object Programming language. The qualitative research was produced with secondary data. The article shows a theory revision of of the Web Systems with service/client technology, the available technologies for development of WBS and also presents concepts and the main diagrams from UML modeling techniques. Finally, it presents a case study of an electronic

journal available at the internet. Main problems found in the development of this electronic journal and UML diagrams examples are presented. The results of the research show that the diagrams of UML presented in relation to the journal consist as a first step to the functionalities development for various actors involved in the system.

Keywords
Web Systems – UML Modeling – Electronic Journal

Introdução

A Tecnologia da Informação (TI), de uma forma abrangente, é a coleção de Sistemas de Informação usada por uma organização para atingir os seus objetivos. Os sistemas têm que estar ligados ao processo de negócio pois a forma como esses sistemas funcionam afeta diretamente o desempenho das organizações nas áreas em que atuam. Neste contexto, nota-se cada vez mais a importância de que os gestores, que demandam sistemas da área de TI das organizações, tenham conhecimento das técnicas de modelagem conceitual e física de projetos de sistemas de informações. A modelagem é a representação da solução de um problema de forma que outros entendam, enquanto o modelo tem a função de comunicação de idéias, conceitos e concepção do que se quer do sistema. A UML – *Unified Modeling Language* é uma linguagem comumente usada para comunicar o modelo em sistemas *Web* Orientado a Objetos. O presente artigo procura apresentar apresentar alguns exemplos de diagramas estáticos e dinâmicos usados na linguagem UML.

Em um primeiro momento, é realizada, por meio de pesquisa bibliográfica e na internet, uma revisão da teoria que aborda a descrição dos sistemas baseados na *Web*, do funcionamento da tecnologia cliente/servidor, uma exposição das tecnologias disponíveis para desenvolvimento de sistemas *Web* e são apresentados conceitos da UML e de seus principais diagramas. O artigo ainda apresenta um estudo de caso qualitativo de uma revista eletrônica que publica artigos da área de Administração impressos e na internet. É exposto o processo de desenvolvimento da revista eletrônica e são apontados alguns problemas encontrados no seu desenvolvimento, informados pelos gestores e programadores. Por fim, são apresentados exemplos de diagramas UML que auxiliariam o desenvolvimento e adequação desta revista para um sistema dinâmico orientados a objetos.

1. *Web-Based Systems* (WBS) - Sistemas *Web*

O princípio do desenvolvimento de sistemas de informação ocorreu basicamente por meio de sistemas em que os dados, os recursos e o processamento eram centralizados, no que normalmente entendia-se por computadores *mainframes*. Neste tipo de sistema havia certa simplicidade de desenvolvimento e de operação, contudo, apresentava pouca flexibilidade para atendimento aos seus usuários. Com a evolução tecnológica os sistemas passaram a ter caráter distribuído, com a execução dos processos espalhada em diversos pontos da rede. A disseminação da execução dos processos na rede permitiu certa otimização do hardware e igualmente uma maior flexibilidade para os sistemas atenderem as necessidades dos usuários. Entretanto, o desenvolvimento comercial da Internet fez aparecer uma nova tendência de centralização da execução dos processos a partir da tecnologia cliente/servidor.

A tecnologia cliente/servidor divide a computação em rede em dois elementos básicos principais para o compartilhamento dos recursos, um computador cliente (que normalmente é um microcomputador) e um computador servidor (que geralmente tem alta capacidade de processamento). Os dois componentes são ligados pela rede física de comunicação, que faz a conexão entre o computador cliente e o servidor. A distribuição da carga de trabalho referente a cada um dos computadores, cliente ou servidor depende do tipo da aplicação que é desenvolvida. No entanto, é trivial se deparar com os trabalhos na maioria dos sistemas concentrados nos computadores servidores, enquanto a tarefa de interface com o usuário fica a cargo dos computadores clientes (TURBAN *et al*, 2003). O computador cliente é o responsável pelo gerenciamento da apresentação, pela lógica básica do aplicativo e pela execução de aplicativos de produtividade pessoal, enquanto o computador servidor fica responsável pelo gerenciamento do acesso, pelas funções de banco de dados, pela execução das regras do negócio e pelo processamento das transações do aplicativo. A rede tem o papel de infra-estrutura para os aplicativos cliente/servidor, pois submete solicitações do cliente para o servidor e transporta os dados resultantes do servidor para o cliente. O sistema

cliente/servidor inclui o hardware (servidor, estações de trabalho e a rede de comunicação) e o software (sistemas operacionais para computadores clientes e servidores, aplicativos do usuário no computador cliente e aplicativos nos computadores servidores).

O conceito da arquitetura cliente/servidor tem dominado a arquitetura de TI das organizações nas últimas décadas, e é a tecnologia base para o desenvolvimento de *Web-Based Systems* (WBS), sistemas baseados na estrutura da Internet. Turban *et al*, 2002 sugerem que as aplicações cliente/servidor estruturadas estão se tornando obsoletas devido ao rápido crescimento destes tipos de sistemas apoiados pela *Web*. Cabe aqui lembrar a distinção entre os conceitos de Internet e de *Web*, comumente confundidos. Por sua fácil acessibilidade, a *Web* se tornou bastante popular, ao ponto de seu conceito se tornar indistinto com o próprio conceito de Internet (TREESE e STEWART, 1998). A Internet é a rede física de todos os computadores conectados a partir de um protocolo comum, o TCP/IP (*Transmission Control Protocol e Internet Protocol*) que possui uma gama de serviços disponíveis, dentre eles, o serviço de páginas de hipertexto conectadas, o que comumente se entende por *World Wide Web* (WWW), ou simplesmente, *Web*. As crescentes popularidade e utilização comercial da Internet podem ser atribuídas ao sucesso da *Web* com os usuários, uma vez que a tecnologia por trás de seus sites é suficientemente simples para a navegação de grande maioria das pessoas (LAUDON e LAUDON, 1999) Tecnicamente, o WBS se refere àquelas aplicações que são contidas em um servidor e podem ser acessadas por meio de um navegador *Web*, portanto, têm a premissa de utilização do protocolo da Internet como rede principal de comunicação entre o computador cliente e o computador servidor. Embora o WBS também seja baseado na tecnologia cliente/servidor, sua implementação é mais barata do que muitos sistemas estruturados que utilizam a mesma tecnologia e, além disso, a conversão de sistemas legados para sistemas baseados em Internet pode ser bem mais fácil e rápida (TURBAN *et al*, 2002).

A premissa de qualquer sistema *Web*, e da própria *Web* em si mesma, é funcionar a partir do conceito cliente/servidor. Os servidores respondem às solicitações dos computadores clientes por páginas *Web* nele contidas, localizadas na rede mediante o endereço da referente página (por meio de uma instrução URL – *Uniform Resource Locator).* Cada servidor da *Web* possui um endereço IP (*Internet Protocol*) exclusivo, por meio do qual este é identificado. Servidores de DNS (*Domain Name Service*) fazem automaticamente a tradução entre a URL e o endereço IP. O HTTP (*HyperText Transfer Protocol*) é o protocolo que permite a comunicação entre os computadores servidores e clientes em sistemas *Web*. Normalmente, um sistema *Web* com a tecnologia cliente/servidor apresenta três camadas principais de aplicação, uma camada de apresentação, uma de regras de negócios e outra de gerenciamento de dados. A camada de apresentação é a responsável pela coleta e apresentação de dados ao usuário, o que inclui entrada e apresentação de dados, formatações, validações e fluxos de navegação. As regras do negócio são as regras do mundo real que devem ser identificadas para garantir a coerência das informações. Essas são as regras que vão formar a base do aplicativo no servidor, coração de um sistema *Web*. Essas regras estão vinculadas ao mundo real. São as diretrizes políticas, legais e comerciais que são intrínsecas à organização, e que, desta forma, devem estar contidas nos sistemas de informação *Web*. A camada de regras é responsável pelo processamento dos dados e pelo cumprimento das regras do negócio, independentemente da interface do usuário e da forma de armazenamento dos dados. A camada de dados é a camada responsável pelo armazenamento, atualização e recuperação dos dados que serão usados no aplicativo. De acordo com Zaneti Jr. e Vidal (2006) o uso de sistemas *Web* ainda está em um estágio inicial. É interessante notar que nesta fase inicial a maioria dos sistemas baseados na *Web* já produzidos têm o foco do desenvolvimento normalmente na interface com o usuário e no aspecto visual do site. Há uma preocupação excessiva com a camada de apresentação em detrimento das camadas de regras do negócio e de banco de dados. Contudo, os sites estão

deixando de ser apenas páginas estáticas de hipermídia interligadas para se tornar sistemas de informação com acesso a banco de dados, portanto, existe uma real necessidade de integração dos sistemas *Web* com os sistemas legados das empresas (VIDAL, 2003). Nota-se deste modo uma relativa importância do conhecimento, por parte dos gestores de negócios, dos processos de modelagem dos sistemas *Web* e das diversas tecnologias que podem suportar o próprio negócio em tais sistemas.

2. Tecnologias para Desenvolvimento de Sistemas *Web*

A tecnologia básica de *design* para sistemas *Web* é a tecnologia de linguagem de marcação de hipertexto denominada HTML *(Hypertext Markup Language).* O HTML é uma linguagem usada para criar e exibir páginas de hipertexto na *Web* relativamente simples, pois usa um conjunto de marcadores (*tags*) para identificar as partes do documento e definir seu conteúdo. As páginas *Web* podem ser ou estáticas ou dinâmicas (ativas). Páginas estáticas podem ter *hyperlinks* para outras páginas e arquivos, mas não atualizam dados no servidor e não são atualizadas pelas ações ou informações do usuário sobre o servidor. Uma página dinâmica ou ativa fornece meios para o usuário interagir com o servidor *Web*, atualizando ou alterando dados no servidor. Normalmente o código de páginas *Web* produzido somente com HTML fornece páginas estáticas. A maioria das páginas da *Web* é realizada com HTML puro, e desta forma são estáticas. Contudo, o interessante para sistemas *Web* é a produção de páginas ativas e dinâmicas. Esta automação pode ser realizada tanto no computador cliente quanto no computador servidor. O código executado no cliente (*client-side code*) é um modo de se automatizar páginas estáticas da *Web*, pois o código do programa é descarregado do servidor *Web* e executado no navegador do computador cliente, o que produz um melhor desempenho do aplicativo.

Um enriquecimento do HTML em busca da dinamicidade de páginas *Web*, a partir da execução do código no computador cliente, é o HTML Dinâmico – DHTML (Dynamic *Hypertext Markup Language)*, que produz textos, gráficos e *hyperlinks* dinâmicos e interativos em páginas *Web*. O DHTML é um conjunto de recursos, baseados em quatro fundamentos: a programação HTML, as definições de Folhas de Estilos em Cascata (CSS – *Cascading Style Sheet*), a tecnologia de Modelo de Objeto de Documento e a Linguagem de Programação *JavaScript* (ROUYER, 1999). Esses fundamentos em conjunto são suportados pelos modernos navegadores, fato que permite a criação de páginas *Web* dinâmicas e interativas. O DHTML funciona a partir da modificação e formatação do documento original com base nos recursos dos navegadores presentes no computador cliente. Uma grande evolução obtida com a programação em DHTML é o tratamento de conjunto de elementos dos documentos como grupos, comandados a partir de ações do usuário. Além do uso do DHTML, a automação de páginas *Web* também pode ser realizada por meio de controles ativos (componentes) ou programas em linguagem de *scripts*[1], o que fornece à página capacidades de processamento de informações. A programação no cliente através de linguagens *scripts* desempenha um papel importante na criação de páginas *Web* com conteúdo ativo, uma vez que com o seu uso, é possível criar páginas *Web* ativas e dinâmicas que interagem com o usuário.

O desenvolvimento de *scripts* é baseado no modelo de programação orientada para objetos, que permite escrever códigos associados a objetos específicos em um aplicativo. Os componentes ativos em um computador cliente são os chamados *applets*, programas reutilizáveis que são executados em navegadores, e os controles *ActiveX*, objetos ou componentes passíveis de inserção em páginas *Web* (ALTER, 2002). Os *scripts* e os componentes ativos no computador cliente são apropriados somente para formatações

[1] *Scripts* são arquivos contendo linguagem interpretada que precisam de softwares interpretadores porque são blocos de códigos não compilados, que são interpretados no tempo de execução

dinâmicas, pequenas consistências de telas e algumas validações de entrada de dados. O uso de linguagem de programação no computador cliente não é adequado para aplicações voltadas a negócios em sites da *Web*, uma vez que pode existir incompatibilidade na interpretação da linguagem utilizada no *script* entre os navegadores disponíveis no mercado e o código do programador fica exposto ao usuário.

A programação no computador servidor é a forma adequada para se tratar a camada de regras de negócio e acessar a camada de dados para WBS. Existe desta forma uma programação embutida em uma página *Web* executada pelo computador servidor quando um computador cliente faz qualquer solicitação. O programa executado no servidor trabalha com o banco de dados, executa as regras de negócio estabelecidas e produz como resultado final uma página em HTML, que é compatível com qualquer navegador e que será enviada ao computador cliente. Esta página enviada ao cliente é uma página dinâmica, pois seu código HTML foi criado dinamicamente pelo computador servidor. Ao contrário das páginas estáticas, a página dinâmica não está fisicamente armazenada no servidor, uma vez que ela é dinamicamente criada para cada solicitação. O processamento no servidor é vantajoso para os WBS, pois oferece fácil acessibilidade, facilidade no gerenciamento, na atualização e na disseminação do sistema por toda a rede Internet proporcionando-o escalabilidade. Contudo existem problemas com relação à programação no servidor, como a sobrecarga do servidor quando houver excesso de usuários e requisições, uma maior vulnerabilidade quanto a questões de segurança e a dificuldade para o controle do que o usuário está realizando com o uso do aplicativo (controle do estado da aplicação).

Existem diversas ferramentas e linguagens utilizadas na criação de páginas *Web* ativas com execução no servidor. Algumas tecnologias utilizadas são: Common Gateway Interface (CGI), Perl (*Practical Extract Report Language), Active Server Pages* (ASP), *Personal Home Page Tools* (PHP), *ColdFusion, J2EE* (Java, Servlets, JSP, JavaBeans, EJB) e *.NET (DotNet) Common Language Runtime*. As plataformas .NET (de propriedade da Microsoft) e a plataforma aberta Java (J2EE) são exemplos de plataformas mais utilizadas atualmente no desenvolvimento de sistemas *Web* Orientado a Objetos. Ambas as plataformas utilizam programação Orientada a Objetos (OO) com o conceito de componentes e camadas, que podem ser desenvolvidas separadamente, desde que as interfaces entre as camadas permaneçam constantes. Esse conceito fomenta o surgimento de novos componentes que se apóiam o mínimo possível em linguagens *script*, ainda necessárias para as chamadas dos componentes, modelo este que tem se mostrado efetivo na construção de sistemas *Web*.

Uma metodologia de desenvolvimento de sistemas é considerada Orientada a Objetos se ela orienta a construção de sistemas a partir do entendimento do mundo real como um conjunto de objetos que se comunicam entre si de forma coordenada. O conceito de programação Orientada a Objetos está ligado à interação de um conjunto de objetos de *software* (programas) que apresentam características únicas e interagem entre si formando um sistema de informação. Cada objeto possui uma série de propriedades e métodos. Um sistema orientado a objetos difere dos sistemas tradicionais de programação linear, pois estes executam processos sobre dados. As principais atividades de uma metodologia Orientada a Objetos consistem em entender quais são os objetos envolvidos no domínio do problema, como eles se comunicam no mundo real e projetar a forma como devem ser implementados. A análise e o projeto de *softwares* Orientados a Objetos definem uma nova modo de pensar nos problemas, pois são utilizados modelos que surgem a partir de conceitos do mundo real. O componente fundamental deste tipo de análise é a classe de objetos, que mescla estrutura e comportamento em uma única entidade que resulta em várias instâncias de objetos semelhantes. A tecnologia de programação OO tem como proemissa a modularidade de seus componentes que podem ser integrados de um modo distinto em alguma outra parte do sistema aplicativo final. As classes de objetos (base da programação OO) descrevem um

grupo de objetos com propriedades e comportamentos semelhantes. Elas contêm informações sobre como uma subclasse ou um objeto deve ser e como estes devem se comportar, definindo portanto a forma e o comportamento padrão de subclasses e objetos nela baseados, portanto, cada objeto é uma instância de uma classe.

O paradigma dos sistemas *Web* com programação Orientada a Objetos revela uma maior importância quanto ao projeto conceitual e físico dos sistemas. O entendimento de como o negócio deve ser transposto no sistema *Web* é essencial para o bom desempenho do sistema e atendimento das funções requisitadas pelos usuários. A arquitetura de um sistema *Web* é a organização lógica em camadas que tem o objetivo de tratar das funções dos aplicativos. A preocupação com a arquitetura de sistemas é importante especialmente quando se trata de Sistemas *Web* Cliente/Servidor, pois facilita a organização dos componentes do aplicativo, quando necessária. A arquitetura do sistema *Web* pressupõe a definição da interface com o usuário, a distribuição de dados e processos, a divisão da aplicação em camadas e componentes, a arquitetura da rede e das comunicações. A arquitetura em camadas é essencial para projetos orientados a objetos, pois os os aplicativos se tornam flexíveis e podem ser alterados com facilidade para atender às reais necessidades do negócio. A arquitetura de um sistema *Web* é abordada por meio da documentação gerada por linguagens de modelagem de sistemas, como a UML – *Unified Modeling Language*.

3. Modelagem de Sistemas *Web*

A modelagem de sistemas tem a função de comunicação da concepção e dos conceitos do sistema, de forma que outros os compreendam. No processo de desenvolvimento de softwares são utilizados modelos para expressar tanto o domínio do problema (projeto conceitual e especificação de requisitos) quanto o domínio do desenvolvimento (projeto físico e de construção). A primeira etapa para o desenvolvimento de um sistema é a especificação de seus requisitos, obtida na fase de levantamento, junto aos usuários. Com os requisitos definidos, é possível a elaboração dos projetos conceitual e físico do sistema. O projeto conceitual define o domínio do problema que o sistema pretende resolver e sua abrangência. Este projeto envolve a modelagem de processos do negócio, dos dados e das regras de negócio. O projeto físico retrata os modelos relativos ao projeto do *software* e à sua construção. A UML – *Unified Modeling Language* é uma linguagem que expressa um conjunto padrão de melhores práticas que apresenta técnicas utilizadas para a expressão de modelos de projetos lógicos e físicos de sistemas, especialmente os orientados a objetos.

Apesar de ter sido concebida para especificação de sistemas de software, esta linguagem também é utilizada para modelar outros tipos de sistemas, como *workflow*, estrutura e comportamento de dispositivos eletrônicos e design de *hardware* (MARTINS, 2002). Ela é uma linguagem de modelagem, com notação gráfica, que especifica, visualiza e documenta os componentes de um sistema, que podem ser respresentados por diagramas distintos (RUMBAUGH *et al*, 1999; MARTINS, 2002), pautada nos relacionamentos básicos estruturais de sistemas orientados a objetos, como a herança, polimorfismo, agregação, generalização e classe (PESSOA, 2000). O objetivo central do desenvolvimento da UML é tentar transformar o foco do desenvolvimento de sistemas do código de programação para a modelagem e, automaticamente, manter a relação entre os dois, ou seja, o programa e o modelo serem entendidos em conjunto, e não separadamente (SIEGEL, 2005).

A notação da UML antes de mais nada deve ser útil para o projeto de desenvolvimento do sistema. Ela deve ser uma ferramenta de ajuda para adequar o negócio em si ao sistema, ou seja, ao código executável dos componentes do sistema. A modelagem com UML exige um alto poder de abstração. A abstração no processo de modelagem é um modo efetivo de se lidar com a sobrecarga na capacidade cognitiva do ser humano de compreender a complexidade de muitos dos atuais sistemas de informação baseados na *Web* (SELIC, 2004). A modelagem

facilita a compreensão da estrutura dos componentes do sistema. É patente que uma modelagem deve estar acompanhada de um claro entendimento do processo de negócio que será retratado no sistema, portanto, é indispensável uma prévia compreensão do conjunto de processos operacionais e gerenciais, e suas respectivas etapas e atividades que formam o negócio em si da empresa ou organização. A UML é uma linguagem de modelagem que utiliza basicamente notação gráfica por meio de diagramas para expressar os seus modelos de projetos. Seus diagramas apresentam tanto a estrutura estática (arquitetura) dos sistemas quanto sua funcionalidade mediante os diagramas dinâmicos. Os diagramas de análise estática descrevem as estruturas e os relacionamentos entre os objetos do domínio do negócio. A análise dinâmica na modelagem revela o comportamento dos objetos em termos de suas mudanças ao longo do tempo. Sua última versão (UML 2.0) define treze tipos de diagramas divididos em três categorias principais, sendo que seis diagramas representam a estrutura estática da aplicação, três representam tipos gerais de comportamento e outros quatro representam diferentes aspectos de interação. Os Diagramas de Estrutura incluem o Diagrama de Classes, Diagrama de Objetos, Diagrama de Componentes, Diagrama de Estrutura, Diagrama de Pacotes e Diagrama de Distribuição. Os Diagramas de Comportamento incluem o Diagrama de Casos de Uso (usado também durante a especificação de requisitos), Diagrama de Atividades e Diagrama de Estados. Os Diagramas de Interação, todos derivados de um generalista Diagrama de Comportamento, incluem o Diagrama de Sequencia, Diagrama de Colaboração, Diagrama de Tempo e Diagrama de Interatividade (SIEGEL, 2005). Os principais diagramas da linguagem UML são os Diagramas de Casos de Uso (modelo que representa a funcionalidade do sistema), os Diagramas de Classe (Modelo Estático) e os Diagramas de Estados, de Seqüência e de Colaboração (Modelo Dinâmico). Em relação ao Modelo de Implementação do software, ainda se destacam os Diagramas de Pacotes, de Componentes e de Implantação, que mostram como o sistema será construído e implementado (RUMBAUGH *et al*, 1999; MARTINS, 2002; PESSOA, 2000).

Uma vez identificados os processos de negócios da empresa, deve se fazer uma análise desses processos e de seus casos de uso, o que o usuário faz ou deseja do sistema *Web*. Os Diagramas de Casos de Uso descrevem a relação entre os atores ligados ao negócio e o sistema em si, e as funções que o sistema deve realizar nestas relações. O sistema vai ser construído a partir da implementação de casos de uso especificados para as interações entre atores e sistema. O Diagrama de Classes é um dos mais importantes componentes da linguagem UML, uma vez que apresenta a estrutura estática de classes do sistema, que reflete como funciona o mundo real e a implementação dos componentes de software a partir das classes estabelecidas. Ele é necessário para identificar os objetos do sistema e suas associações. Neste ponto, é essencial descobrir quais são os objetos do mundo real que serão representados no sistema como classes de objetos, com quais outras classes essa classe se relaciona e quais são as classes de objetos necessárias para cada caso de uso.

Em um segundo momento, a UML é útil para apontar a colaboração entre as classes dentro dos casos de uso, (Diagramas de Interação) e como os objetos se comportarão no tempo (Diagramas de Estados). Os Diagramas de Interação refletem como as classes se interagem em algum comportamento do sistema. Os Diagramas de Interação podem ser Diagramas de Seqüência ou Diagramas de Colaboração. Os Diagramas de Seqüência apresentam o fluxo de interações entre os objetos no tempo. Os Diagramas de Colaboração refletem como objetos são ligados entre si com a apresentação dos papéis de colaboração entre os objetos. Os Diagramas de Estados refletem como um objeto se comporta no tempo, entre os diversos Casos de Uso. Este diagrama apresenta a seqüência de estados que um objeto pode assumir em resposta a eventos. Outros diagramas em destaque são os Diagramas de Atividades, de Pacotes, de Componentes e de Implantação. Os Diagramas de Atividades apresentam a estrutura de controle das atividades descrevendo as atividades exercidas em um

processo. Os Diagramas de Pacotes, que apresentam os grupos de classes e as dependências entre elas, os Diagramas de Componentes, que apresentam as dependências entre os componentes do sistema e os Diagramas de Implantação, que apresentam a configuração de processamento e os componentes do sistema são os diagramas que refletem a estrutura de funcionamento do sistema.

A mudança de paradigma de programação de sistemas tradicionais para sistemas Orientados a Objetos mostra a importância da modelagem prévia do sistema, mediante o uso dos diversos diagramas contidos na linguagem UML. A seguir é apresentada uma pesquisa feita por meio de um estudo de caso de implantação de uma Revista Eletrônica disponível na Internet, com análise da tecnologia utilizada e de como a modelagem UML é capaz de potenciar a implantação de um sistema *Web* que tornaria a revista dinâmica, do ponto de vista de processos de negócios.

4. Estudo de caso - Revista Eletrônica

4.1 Objetivos e Metodologia de Pesquisa

O objetivo da presente pesquisa é apresentar à luz da teoria da modelagem UML, como um processo de negócio de um sistema de revista eletrônica disponível na Internet poderia ser representado em relação aos principais diagramas UML. Esta pesquisa tem natureza exploratória, com análise qualitativa dos procedimentos de desenvolvimento dos implantadores e de demanda dos gestores do site da revista eletrônica até o mês de dezembro de 2006. Mediante informações dos gestores e programadores da revista, serão expostas as tecnologias utilizadas para confecção do sistema, bem como os principais problemas encontrados nos sistemas na visão destes atores. Em relação a um processo de negócio não atendido pelo sistema, dentre todo o rol de problemas encontrados, será exemplificado por meio dos diagramas UML como o sistema poderia ser melhorado com a modelagem UML e posterior implementação de programação Orientada a Objetos em um sistema *Web*.

A Revista Eletrônica, objeto do presente estudo de caso, foi escolhida para esta pesquisa por ser um caso típico de programação de sistemas *Web* com páginas estatísticas e tecnologia HTML preponderante. A revista é editada quadrimestralmente em formato impresso e reproduzida no site em formato eletrônico. A gerência da divulgação eletrônica da revista é feita por um editor-gestor e um programador. Este periódico pertence a uma instituição de ensino superior da cidade de Franca, no interior do Estado de São Paulo, e está no ar desde o mês de janeiro do ano de 2004.

É um periódico divulgador de discussões nacionais e internacionais no escopo das ciências administrativas com vistas a socializar o conhecimento na área de administração. A revista aceita a submissão de artigos para serem publicados, que serão avaliados por especialistas em forma *blind review*, com posterior divulgação da aprovação ou reprovação para publicação. É possível a submissão de artigos acadêmicos nas línguas portuguesa, francesa, inglesa e espanhola. Os artigos enviados para submissão são recebidos por meio de mensagens eletrônicas (e-mails). Existe um endereço de e-mail específico para o recebimento de artigos submetidos. Os artigos devem ser submetidos como arquivos em formato Word, anexados à mensagem de e-mail, de acordo com normas de publicação pré-estabelecidas.

4.2 Implantação e Problemas Atuais da Revista Eletrônica

A implantação da primeira versão on-line da revista eletrônica ocorreu nos primeiros meses de 2004. Toda a programação da revista foi feita exclusivamente com tecnologia HTML e DHTML. Foi utilizado um editor de HTML comum no mercado, que utiliza recursos básicos de DHTML, com biblioteca de arquivos que geram efeitos de páginas dinâmicas no decorrer da navegação. Contudo, todas as páginas da revista são estáticas. A atualização do site da revista ocorre quadrimestralmente, assim que uma nova edição impressa é publicada,

automaticamente esta edição é transposta para a versão eletrônica da revista. Os arquivos dos artigos publicados são apresentados em formato pdf (*portable document file*), protegidos com senha. Contudo, a programação de acesso aos artigos publicados e de busca de artigos foi realizada sem acesso automático a banco de dados. Todos os arquivos estão organizados em estrutura de pastas, sem a utilização de sistema de banco de dados para acesso. Alguns dos principais problemas encontrados na versão eletrônica da revista, segundo gestores e programadores, são:

- Pouca atualização do site. Versão estática dificulta a atualização, apesar do uso de DHTML e de arquivos CSS e do conceito de objetos em menus, cabeçalhos, rodapés e títulos;
- Inexistência de sistema de busca de artigos publicados por palavras-chave no título, autor, assunto ou data; o sistema de busca presente baseado nas edições é precário e não facilita a busca de um artigo específico pelo usuário;
- Módulos de versão em línguas estrangeiras, inglês, francês e espanhol ainda não haviam sido implantados;
- Ausência de controle de tráfego sobre o site da revista. A não obrigatoriedade de um cadastro de identificação não permite que os gestores tenham um controle efetivo sobre os usuários que navegam pelo site da revista.
- Ausência de um sistema de recebimento e controle de artigos submetidos para avaliação. Os artigos são recebidos por meio de e-mails, exigindo uma pessoa para a elaboração da triagem e confirmação ao autor de recebimento do artigo para avaliação; inexistem ainda no sistema recursos de envio automático dos artigos submetidos aos avaliadores;
- Inexistência de um sistema de relacionamento do periódico com os avaliadores, em que seria possível o controle de números de artigo enviado para cada avaliador, temas de interesse do avaliador e controle de prazos, datas de entrega e de status de avaliação dos artigos avaliados pelos diversos avaliadores.
- Inexistência de um sistema on-line de contato do usuário com os diversos setores da revista. O contato é feito exclusivamente por meio de mensagens eletrônica. Ausência ainda de páginas de resolução de dúvidas freqüentes.

Estes são alguns problemas apresentados que mostram a pouca utilidade do atual sistema implantado para o negócio da revista eletrônica em si, ou seja, como fomentador de eficiência de processos do funcionamento da revista. A maioria desses problemas ocorreu devido à falha na definição dos processos de negócio iniciais da revista e na implementação da tecnologia HTML com foco em páginas estáticas e preocupação exclusivamente com a interface de apresentação e não com o negócio em si.

5. Exemplos de Diagramas UML para o Sistema de Revista Eletrônica

O entendimento de como funciona a programação Orientada a Objetos, e do processo de modelagem de negócios a partir da linguagem UML, permite ao gestor e programador uma visualização mais nítida de como o mundo real da revista poderia ser representado no sistema. A seguir são apresentados os principais diagramas de modelagem para uma melhoria no sistema da revista eletrônica, com vista a solucionar um dos problemas encontrados em relação aos serviços oferecidos pela revista on-line. Pretende-se, como exemplo de modelagem, elaborar os diagramas para a funcionalidade do sistema de recebimento e controle de artigos submetidos para avaliação e do sistema de relacionamento da revista com os avaliadores, e demais aspectos desta relação, como o controle de números de artigo enviados para cada avaliador, temas de interesse, prazos, datas de entrega e status de avaliação dos artigos avaliados pelos avaliadores.

5.1 Diagrama de Casos de Uso

O Diagrama de Casos de Uso é usado para capturar os requisitos funcionais do sistema da Revista Eletrônica com relação ao processo de avaliação. Os Casos de Uso representam a interação entre os atores e o sistema. No exemplo do Diagrama de Casos de Uso são três os atores, o autor que submete o arquivo, o avaliador e o sistema da Revista Eletrônica. Os Casos de Uso mencionados representam unidades de um trabalho significante no sistema da revista e descrevem a funcionalidade que deverá ser construída no sistema proposto. Um Caso de Uso pode "incluir" outra funcionalidade como o caso dos Casos de Uso “Submete artigo” e “Cadastra Autor”.

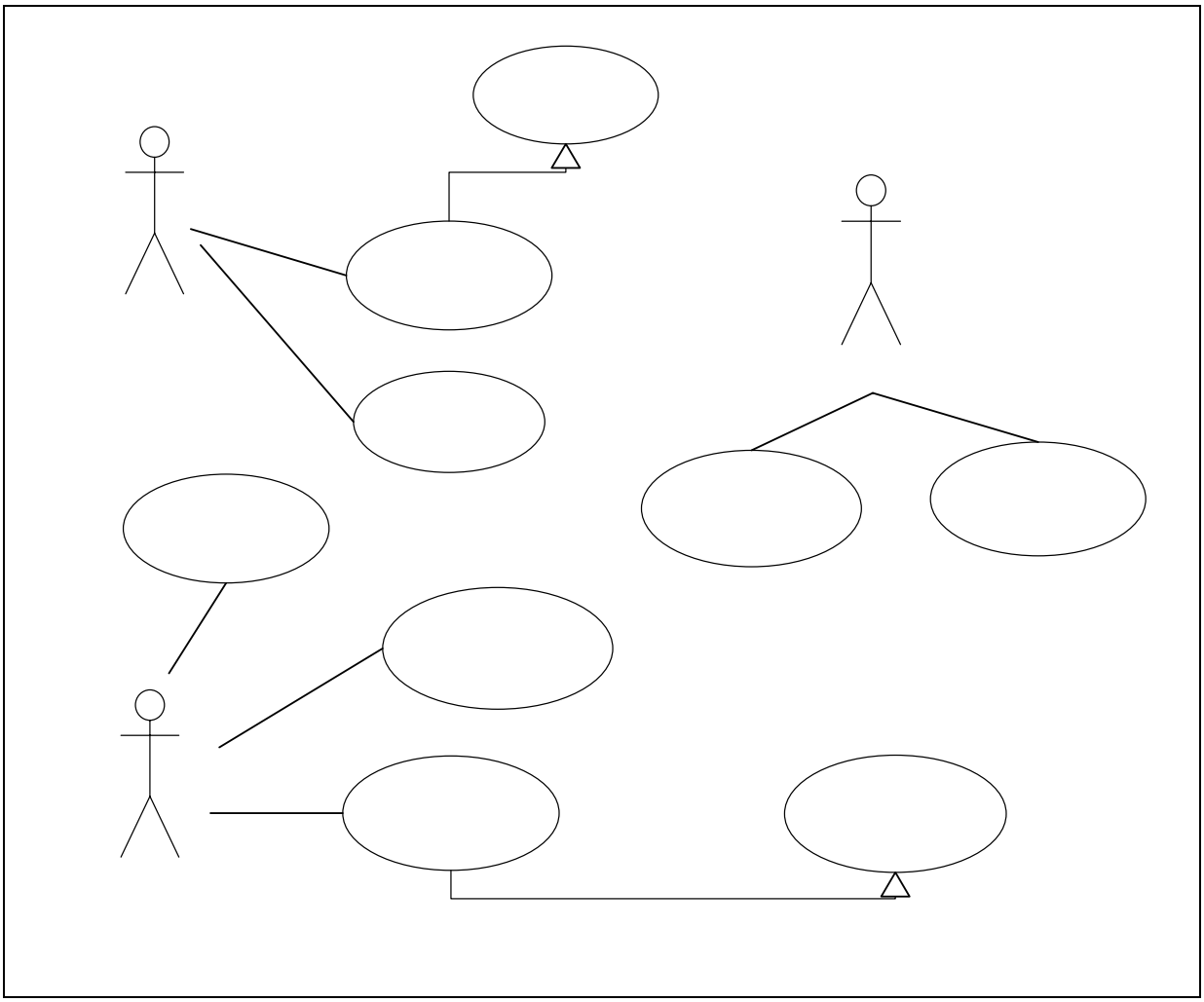


Figura 1: Exemplo de Diagrama de Casos de Uso

5.2 Diagrama de Classes

O diagrama de classes é uma representação da estrutura e das relações das classes que servem de modelo para os objetos da revista eletrônica. É uma modelagem estática muito útil, pois define as classes de objetos necssárias ao sistema e serve como base para a elaboração da modelagem dinâmica, por meio dos diagramas de comunicação, sequência e estados. No exemplo do diagrama de classes para a revista eletrônica estão especificadas nove classes de objetos com suas respectivas relações entre si. Ainda estão listados os atributos que definem as características da classe e as operações e funções requeridas aos objetos.

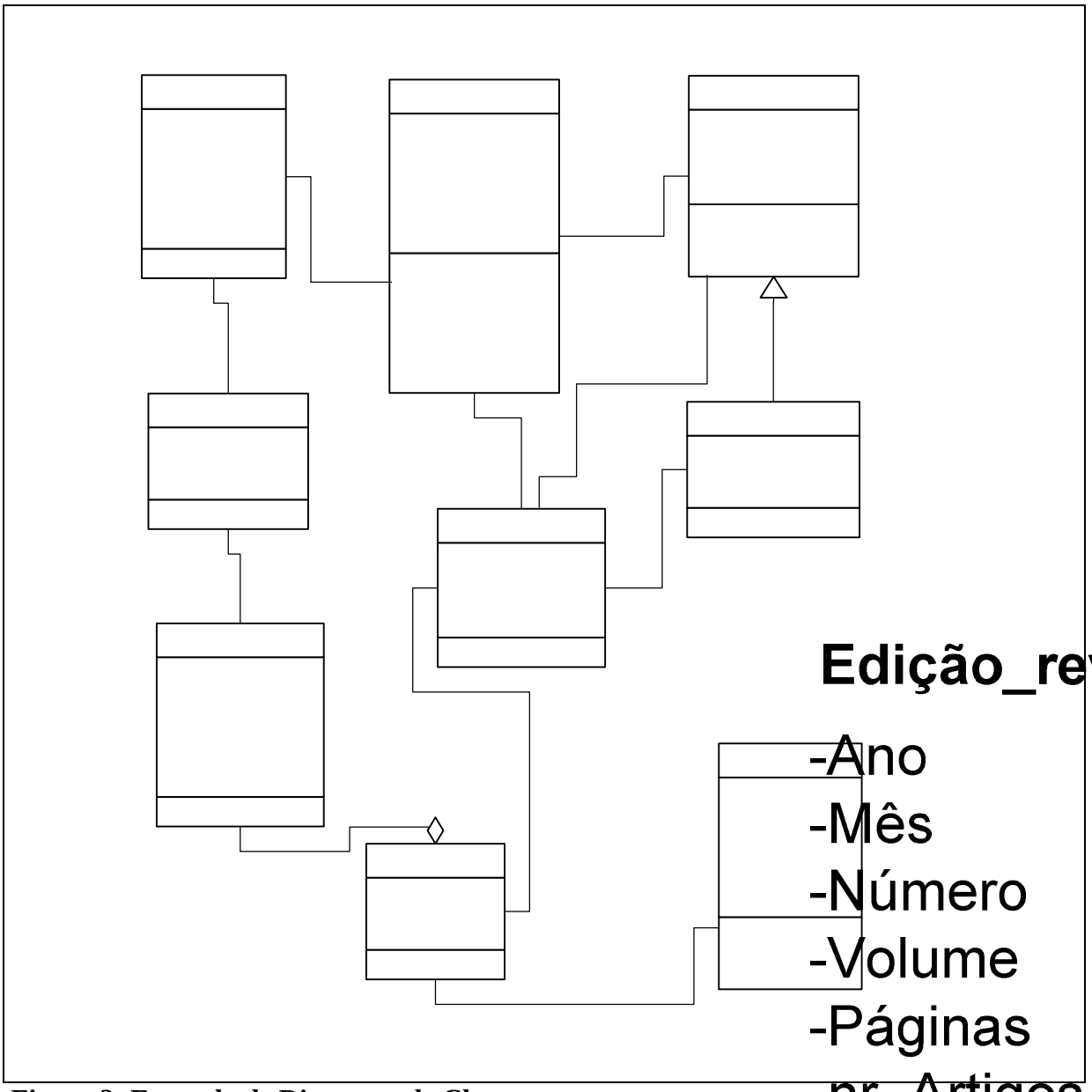


Figura 2: Exemplo de Diagrama de Classes

5.3 Diagrama de Estados

Os Diagramas de Estados mostram como uma classe de objetos se comporta no tempo. Eles representam a seqüência de estados que um objeto pode assumir em resposta a eventos e apresentam o ciclo de vida de uma classe, os eventos que geram a transição de estados e as ações resultantes da mudança de estado. O estado de um objeto é um dos possíveis modos em que o objeto pode existir, condição ou situação durante a vida do objeto na qual ele satisfaz condições, executa atividades ou espera por eventos. No exemplo de diagramas para a revista Eletrônica está representado um Diagrama de Estado descrevendo o comportamento da classe de objetos “Artigo”. O estado inicial da classe é o artigo submetido, e o estado final é o artigo aprovado. Entre os dois estados existem vários outros estados que mostram qual a situação do artigo submetido no exato momento de consulta (em avaliação, avaliado, publicado, etc).

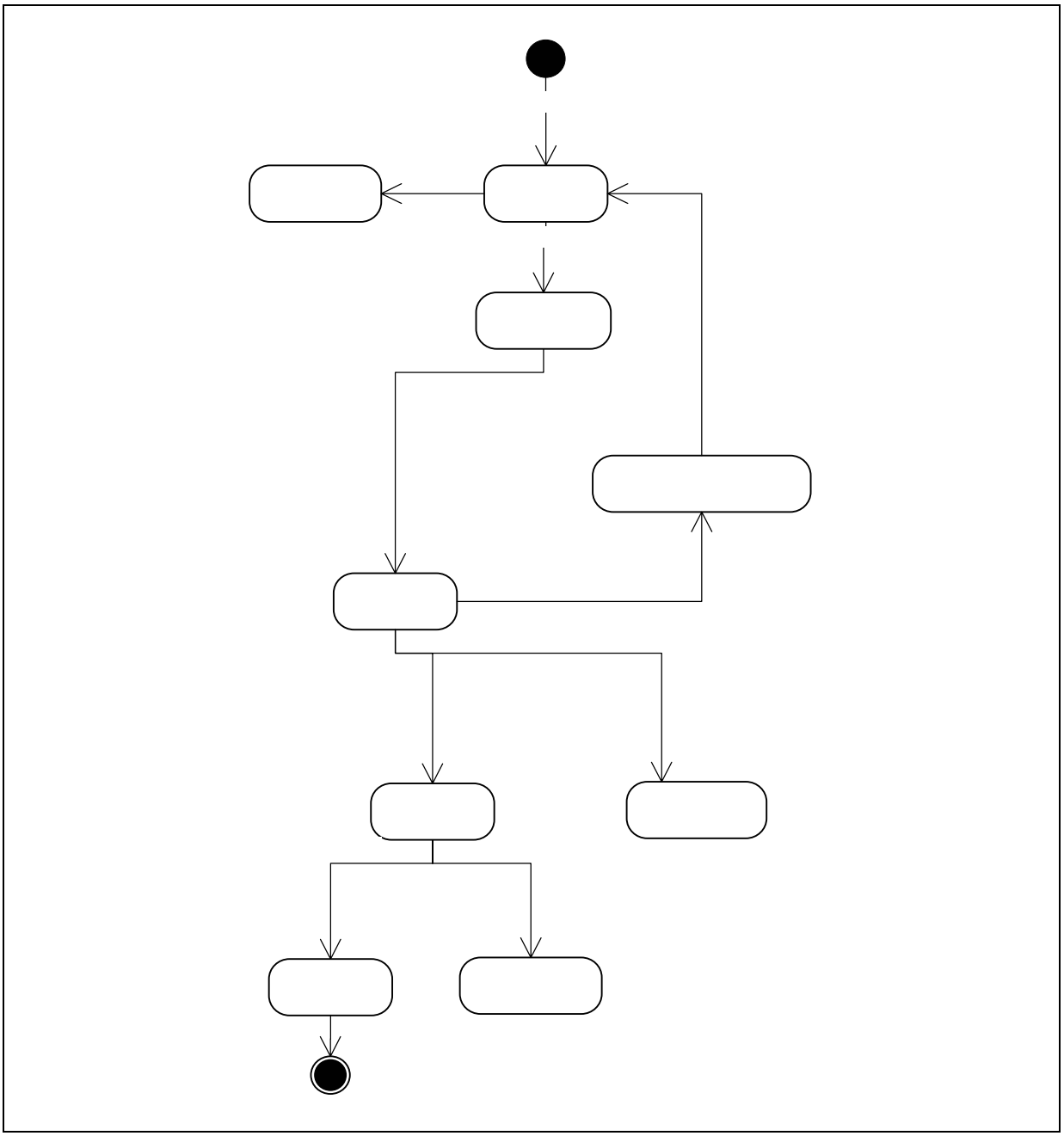
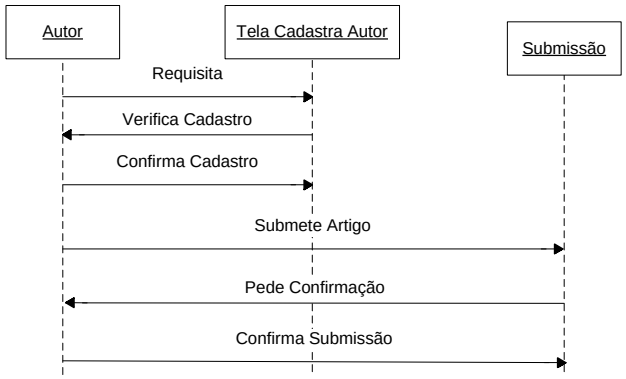


Figura 3: Exemplo de Diagrama de Estados

5.4 Diagrama de Seqüência e Colaboração

Os diagramas de Sequência e de Colaboração são isomórficos, ou seja, há correspondência entre os elementos destes dois tipos de diagramas. O Diagrama de Sequência mostra a sequência de processos (mensagens) que são passadas entre os objetos ao longo do tempo, registrando o comportamento de um caso de uso. Ele exibe os objetos e as mensagens passadas entre esses objetos no caso de uso. O Diagrama de Colaboração exibe a interação entre os objetos, com seus relacionamentos, e também inclui as mensagens que são trocadas entre eles.

Seqüência



Colaboração

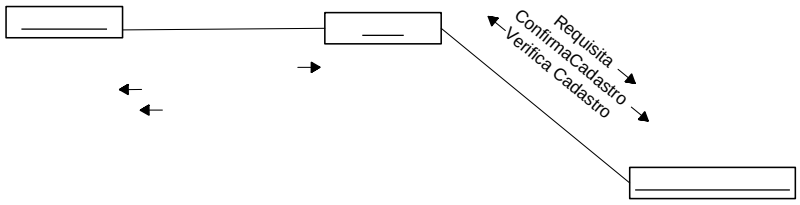
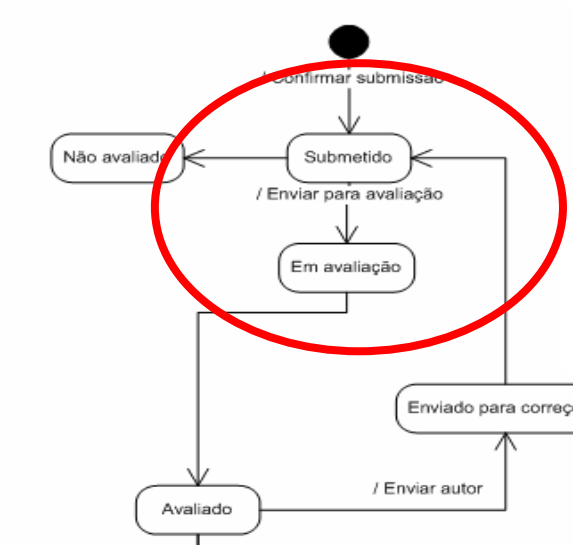


Figura 4: Exemplo de Diagrama de Sequência e Colaboração

6. Soluções Possíveis para a Revista Eletrônica – Seleção de Artigos para Análise de Conteúdo

O conhecimento dos princípios de modelagem permite aos gestores ter a capacidade de conhecer os aspectos inerentes aos sistemas que podem ser desenvolvidos com vistas à solução dos problemas apontados que resultam em pouca utilidade do atual sistema implantado para a revista eletrônica. È importante reforçar a idéia de que o sistema deve sempre existir como suporte do negócio e fomentador de agilidade ao processo, portanto, a correta modelagem é um passo importante para o desenvolvimento de funcionalidades importantes para a eficiência do processo. Como exemplo, é possível citar a atuação do sistema no processo de automação do envio de artigos para avaliação. Um dos principais problemas apontados em relação à revista estudada consiste na ausência de um sistema de controle de artigos submetidos para avaliação e de recebimento e envio automático destes artigos para os respectivos avaliadores, com base na formação acadêmica e da área de atuação dos diversos avaliadores cadastrados. O desenvolvimento dsta funcionalidade no sistema tornaria possível o redirecionamento automático de artigos submetidos a partir da análise de conteúdo do texto do artigo, feita com auxílio de algoritmos próprios para análise de conteúdo, similares aos sistemas utilizados para evitar a propagação de *Spams*. Simultanemante o sistema seria capaz de abranger o controle de números de artigo enviado para cada avaliador, os temas de interesse do avaliador e o controle de prazos, datas de entrega e de status de avaliação dos artigos avaliados pelos diversos avaliadores.

Uma das formas de analisar o conteúdo de sistemas anti-spams é a partir de pesos e regras ou de filtros bayesianos (suportados pela Teoria Probabilística de Bayes). No sistema de regras o conteúdo da mensagem é analisado através de diversas regras compostas por testes lógicos que verificam determinadas características, frases ou palavras que são comumente encontradas em spams ou mensagens legítimas, sendo que para cada regra existem pesos (positivos ou negativos) que indicam a probabilidade da mensagem ser spam ou legítima. Para classificar as mensagens, o sistema aplica todas as regras e classifica a mensagem como spam a partir da soma dos pesos das regras correspondentes. Os filtros bayesianos utilizam uma forma de classificação a partir de Redes Bayesianas, que não levam em conta regras pré-estabelecidas, mas sim probabilidades de artigos serem enquadrados em categorias a partir de artigos já enquadrados previamente (TAVEIRA e DUARTE, 2006). Ambas as funcionalidades dos sistemas são aplicadas em filtros de spams, e poderiam ser aplicadas como filtros para os artigos enviados na revista eletrônica. É óbvio que o processo de agrupamento de artigos nas diversas áreas do pensamento administrativo é bem mais complexo que a separação entre mensagens legítimas ou spams, contudo, um processo de análise de conteúdo permitiria atribuir ao sistema uma funcionalidade que hoje é exclusivamente manual, e que consome tempo e traz ineficácia ao processo de negócio da revista eletrônica.



Área de Atuação - Proposta de Automatização do Envio dos Artigos para os Avaliadores a partir de Análise de Conteúdo

Considerações Finais

O propósito principal deste artigo foi o de apresentar alguns exemplos de diagramas estáticos e dinâmicos usados na linguagem UML, usando como estudo de caso uma revista eletrônica disponível na internet. Foi realizada uma revisão da teoria que mostrou a importância da modelagem UML para a compreensão de sistemas baseados na *Web*. A UML pode ter uma utilização ampla, pois compreende tanto os aspectos estáticos relativos a estrutura quanto os aspectos dinâmicos e as funcionalidades dos sistemas.

Os diagramas apresentados em relação à revista eletrônica formam um passo inicial facilitador para o desenvolvimento de uma revista que tenha funcionalidades para os variados atores. Os diagramas são capazes de comunicar o projeto conceitual do sistema a qualquer pessoa que deseja tomar conhecimento de tais modelos da revista. A linguagem UML pode ser aplicada em diversas fases do desenvolvimento de um sistema *Web*, desde a definição dos requisitos, passando pelas fases de projetto lógico, físico, de construção, implementação e testes, portanto o conhecimento desta ferramenta e de seus diagramas é indispensável para

programadores e aconselhável para gestores, pois permite uma otimização dos esforços em busca de sistemas com alta relação de funcionalidades.

Obviamente, a compreensão das técnicas de modelagem em UML vai muito além deste artigo. Este documento pretende ser apenas um texto instigador que estimule ao gestor ter a iniciativa de aprender sobre técnicas de modelagem que auxiliem no desenvolvimento dos sistemas de sucesso que se deseja estarem presentes nas organizações e empresas. O conhecimento pleno do gestor sobre o que é possível extrair das tecnologias atuais de desenvolvimento de sistemas em conjunto com os princípios de modelagem permitem uma orientação voltada para o máximo aproveitamento de sistemas desenvolvidos com vistas a fornecer agilidade ao processo de negócio pretendido.

Referências

ALTER, S. Information Systems: The Foundation of E-Business. 4th edition. New Jersey: Prentice Hall, 2002.

FACEF PESQUISA, Revista Facef Pesquisa, Disponível em www.facef.br/facefpesquisa Acesso em 20/12/06, 2006.

LAUDON, K.C., LAUDON, J.P. Information Systems and the Internet: A Problem-Solving Approach. Tradução da 4a. edição. Rio de Janeiro: LTC, 1999.

MARTINS, J.C.C. Gestão de Projetos de Desenvolvimento de Software: PMI - UML. Rio de Janeiro: Brasport, 2002.

PESSOA, A. Projetos de Sistemas de Informação: A Visão Orientada a Objetos. Rio de Janeiro: Book Express, 2000.

ROUYER, J. DHTML Efeitos Mágicos; Tradução Roberta Aquino. São Paulo: Quark Books, 1999.

RUMBAUGH J., BOOCH, G., JACOBSON, I., The Unified Modeling Language User Guide. Addison Wesley, 1999.

SELIC, B. UML 2.0: Exploiting Abstration and Automation. IBM Rational Software, Março, 2004. Disponível em http://www.sdtimes.com/editorial/opinion-20040315-01.html Acessado em 28/12/06

SIEGEL, J. Introduction to OMG UML. Object Management Group. Julho, 2005. Disponível em http://www.omg.org/gettingstarted/what_is_uml.htm, Acessado em 28/12/06.

TAVEIRA, D.M. DUARTE, C.M.B. Avaliação de Desempenho de Mecanismos Anti-Spam e Análise de Características de Spams. UFRJ, 2006. Acessado em 10/04/07. Disponível em: www.gta.ufrj.br/ftp/gta/TechReports/TaDu06.pdf

TREESE, G.W., STEWART, L.C. Designing Systems for Internet Commerce. Addison Wesley, 1998.

TURBAN, E., MCLEAN, E., WETHERBE, J. Information Technology for Management: Transforming Business in the Digital Economy. 3rd Edition. Ed. Wiley, 2002.

TURBAN, E., RAINER JR., R.K., POTTER, R.E. Administração da Tecnologia da Informação: Teoria e Prática. Trad. 2a. Ed. Americana. Rio de Janeiro: Campus, 2003.

UML 2.0 - Unified Modeling Language. UML 2.0, The Current Official Version. Disponível em http://www.uml.org/#UML2.0, Janeiro, 2007

VIDAL, A.G.R. Projeto Conceitual de Sistemas de Informação. USP São Paulo: Junho, 2003.

ZANETI JÚNIOR, Luiz Antonio ; VIDAL, A. G. R. . Construção de sistemas de informação baseados na Tecnologia *Web*. RAUSP. Revista de Administração, v. 41, p. 232-244, 2006.