Pós-Graduação em Ciência da Computação

“Uma Extensão do Fluxo de Análise e Projeto do RUP para o Desenvolvimento de Aplicações Web”

Por

Ricardo André Cavalcante de Souza

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://www.cin.ufpe.br/~posgraduacao

RECIFE, ABRIL/2002

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

Ricardo André Cavalcante de Souza

Uma Extensão do Fluxo de Análise e Projeto do RUP para o Desenvolvimento de Aplicações Web

Trabalho apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR:

Prof. Alexandre Marcos Lins de Vasconcelos

Recife, Abril de 2002

Agradecimentos

A Deus nosso criador por me iluminar e permitir mais esta conquista.

Ao meu saudoso pai Manoel André (*in memoriam*), eterno amigo e incentivador, por todos os ensinamentos.

A minha amada mãe Rosilene por todos os sacrifícios e pela educação que me proporcionou. Aos meus irmãos André e Rosiléa pelo carinho e apoio.

A minha Aninha pelo companheirismo, por me ouvir, apoiar e por estar presente nas horas mais difíceis.

Aos meus conterrâneos e grandes amigos Alexandre Macedo e Marco Fagundes que me acompanham desde os tempos da graduação ainda no nosso querido estado do Pará.

A equipe do projeto SIG@UFPE, da qual tenho orgulho de fazer parte, por toda a colaboração e amizade.

Ao professor Alexandre Vasconcelos pela valiosa orientação na elaboração do trabalho. Ao professor Daniel Schwabe pela grande colaboração e avaliação do trabalho. Ao professor Augusto Sampaio pela avaliação do trabalho.

Resumo

A Internet tem se mostrado como um dos mais efetivos e atrativos meios para realização de transações comerciais, possibilitando tanto a divulgação, quanto a negociação e disponibilização de bens e serviços. Além disto, devido em grande parte a globalização da economia mundial, cada vez mais as empresas estão migrando seus sistemas corporativos para plataformas baseadas principalmente na Web. Diante deste panorama, hoje em dia existe uma demanda bastante significativa pelo desenvolvimento de aplicações para Web.

Entretanto, do ponto de vista da Engenharia de Software, as aplicações Web possuem características específicas que as diferenciam das aplicações tradicionais, e que precisam ser tratadas no processo de desenvolvimento de uma forma disciplinada. Para tanto, fazem-se necessárias adaptações em processos de desenvolvimento de software já existentes para um melhor atendimento na construção de aplicações Web.

Visando atender as necessidades de desenvolvimento específicas para aplicações Web, este trabalho apresenta uma adequação da metodologia genérica de desenvolvimento de software RUP (*Rational Unified Process*), mais especificamente no fluxo de Análise e Projeto, tendo em vista que a etapa de análise e projeto é onde há uma maior diferença no desenvolvimento de aplicações Web com relação a aplicações tradicionais.

Este trabalho apresenta também um estudo de caso para validação da efetividade da proposta de extensão do fluxo de Análise e Projeto do RUP para o desenvolvimento mais apropriado de aplicações Web.

Abstract

The Internet has shown to be one of the most effective and attractive ways for the accomplishment of business, making possible diffusion, negotiation and delivery of products and services. Furthermore, mostly due to the globalization of the world's economy, more companies are migrating their corporate systems to Web-based plataforms mainly. In such a panorama, there is a significant Web-based applications development demand.

However, from the Software Engineering point of view, Web-based applications have specific characteristics that must be handled in the development process. For this reason, it is necessary to do some adjustments in existing software development process.

Aiming at the solution of Web-based applications development related issues, this work presents a RUP (Rational Unified Process) Analisys & Design workflow extension and a case study to validate it.

Índice

Índice de Figuras

Capítulo 1

Introdução

O crescimento e a popularização da Internet como meio de comunicação e divulgação de notícias e produtos possibilitou o surgimento de novas formas de transações comerciais e, cada vez mais, a comunidade de negócios tem migrado seus sistemas críticos para uma plataforma baseada na Web, o que tem ocasionado uma grande demanda no desenvolvimento de aplicações Web.

As aplicações Web possuem características específicas que as diferenciam de aplicações tradicionais, e que precisam ser tratadas no processo de desenvolvimento. Diante deste panorama, este trabalho apresenta uma proposta de extensão do fluxo de Análise e Projeto do RUP (*Rational Unified Process*) para o desenvolvimento mais apropriado de aplicações Web.

Neste capítulo, são apresentados a motivação, o escopo e os objetivos deste trabalho, bem como sua estruturação em termos de capítulos e apêndices.

1.1 Motivação

Devido ao grande crescimento da Internet, uma nova forma de comércio surgiu e, cada vez mais, os negócios estão sendo direcionados para o mundo virtual, isso se dá nas mais diversas áreas comerciais, desde livrarias até bancos virtuais. Entre as principais vantagens dessa nova forma de comunicação com o cliente está a desburocratização, a comodidade e a diminuição de custos operacionais. Além disto, as empresas estão migrando seus sistemas corporativos e departamentais para plataformas baseadas na Web, devido em grande parte à globalização da economia mundial, tendo em vista que o espaço geográfico de atuação de uma empresa pode ser em âmbito mundial.

A Internet e, conseqüentemente, as aplicações Web estão a cada dia mais presentes em nossas vidas e, se por um lado essa "popularização" possibilita uma maior disseminação do conhecimento e outras inúmeras vantagens, por outro lado, do ponto de vista da Engenharia de Software, nos deparamos com preocupações que antes não tínhamos com as aplicações tradicionais.

As fases do ciclo de um sistema de informação necessitam ser adaptadas para melhor atenderem o desenvolvimento de aplicações Web, desde os requisitos (com grande atenção aos requisitos não-funcionais), até tratar questões como audiência da aplicação e compatibilidade de ferramentas, além de preocupação excessiva com segurança.

As aplicações Web estão tornando-se cada vez mais complexas devido aos vários aspectos adicionais e específicos que devem ser observados para construção deste tipo de aplicação, além da complexidade natural do negócio. Diante deste panorama, faz-se necessário adaptações em processos de desenvolvimento de softwares já existentes, para melhor atenderem às necessidades do desenvolvimento de aplicações Web.

1.2 Escopo do Trabalho

O enfoque deste trabalho é sobre a Engenharia de Software para Web cujo propósito é o estabelecimento e uso de princípios de engenharia, gerenciamento e técnicas sistemáticas e disciplinadas para o desenvolvimento, implantação e manutenção de aplicações Web de alta qualidade, bem como a incorporação de princípios bem sucedidos da Engenharia de Software tradicional.

Mais especificamente, este trabalho consiste na adaptação da metodologia de desenvolvimento de software RUP – *Rational Unified Process*, para construção de aplicações Web. As vantagens do uso de uma metodologia [Bennett 99] são: ajuda a produzir um produto de melhor qualidade, em termos de padrões de documentação, aceitação do usuário, manutenibilidade e consistência do software; ajuda a garantir que os requisitos do sistema sejam completamente atendidos; ajuda a gerenciar o projeto, pois possibilita um melhor controle da execução do projeto e a redução de custos de desenvolvimento; promove a comunicação entre participantes do projeto, pela definição dos participantes essenciais e interações, bem como pela estruturação do processo como um todo.

A metodologia RUP foi escolhida devido à sua generecidade e por ser adaptável, bem como por reunir o melhor de várias técnicas modernas de desenvolvimento de software e ser bem aceita tanto no meio comercial como no meio acadêmico.

O RUP, como metodologia, atende todo o ciclo de vida do desenvolvimento de software, entretanto, a etapa que mais diferencia o desenvolvimento de aplicações Web com relação as aplicações tradicionais é a de análise e projeto [Araújo 01]. Desta forma, o foco deste trabalho está na extensão do fluxo de Análise e Projeto do RUP para o desenvolvimento mais apropriado de aplicações Web.

A extensão do fluxo de Análise e Projeto do RUP consiste da criação de novas atividades e modificações de atividades já existentes, e visa tratar características específicas de aplicações Web, com ênfase nos aspectos de navegação e de apresentação.

1.3 Objetivos

Este trabalho visa atender o processo de desenvolvimento de aplicações Web que seguem o padrão arquitetural *Layers* (Camadas) – Figura 1-1.

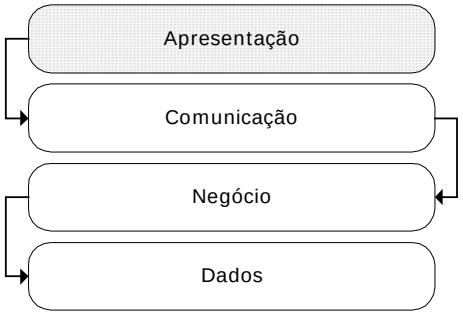


Figura 1-1 Padrão Arquitetural em Camadas

Em [Borba 00], são definidos os propósitos das camadas apresentadas na Figura 1-1. A camada de Apresentação, também chamada Interface com o Usuário, é responsável pela apresentação da aplicação ao usuário. A camada de Comunicação permite o acesso remoto aos serviços da aplicação. A camada de Negócio possui responsabilidades inerentes à aplicação. A camada de Dados é responsável pelo acesso e manipulação de dados, bem como por efetuar o interfaceamento da aplicação com o banco de dados.

A camada de apresentação é a que mais se diferencia entre as aplicações Web e as aplicações tradicionais, devido ao fato de que em aplicações Web alguns aspectos específicos devem ser levados em consideração como a navegação, a organização e apresentação das interfaces gráficas. As demais camadas (comunicação, negócio, dados) de aplicações Web são similares as de qualquer aplicação distribuída.

Desta forma, o objetivo principal deste trabalho é estender o fluxo de Análise e Projeto do RUP para incorporar o projeto da Camada de Apresentação de aplicações Web, que consiste na criação do Modelo Navegacional com o propósito de identificar os caminhos navegacionais da aplicação, ou seja, como o usuário caminha (navega) pela aplicação para utilizar as funcionalidades e, baseado nas necessidades navegacionais, projetar as interfaces gráficas para interação do usuário com a aplicação. Este trabalho também propõe um método para criação do Modelo Navegacional, bem como recomendações para o desenvolvimento mais apropriado de aplicações Web.

A proposta de extensão do fluxo de Análise e Projeto do RUP incorpora algumas atividades do framework de Análise e Projeto para aplicações Web proposto por A.Araújo [Araújo 01] e define novos artefatos com base nas extensões de UML propostas por J.Conallen [Conallen 99b] e por N.Koch [Koch 01].

1.4 Estrutura do Trabalho

Além deste capítulo introdutório, o trabalho consiste de mais sete capítulos e três apêndices.

O Capítulo 2 introduz a Engenharia de Software para Web e apresenta características específicas das aplicações Web que as diferenciam das aplicações tradicionais.

O Capítulo 3 apresenta o estado da arte de extensões de UML para aplicações Web, bem como um estudo comparativo sobre quais aspectos são atendidos por cada uma das extensões de UML apresentadas.

O Capítulo 4 traz um resumo da metodologia de desenvolvimento de software RUP, através da apresentação de suas características, fases e fluxos.

O Capítulo 5 apresenta uma proposta de extensão do fluxo de Análise e Projeto do RUP para o desenvolvimento mais apropriado de aplicações Web.

O Capítulo 6 apresenta o detalhamento das atividades e artefatos do subfluxo proposto para o projeto da Camada de Apresentação de aplicações Web, bem como o método proposto para criação do Modelo Navegacional.

O Capítulo 7 apresenta a validação do subfluxo proposto para o projeto da Camada de Apresentação, através de sua aplicação a um estudo de caso realizado sobre a aplicação Web SIG@UFPE (Sistema de Informações e Gestão Acadêmica da UFPE).

O Capítulo 8 apresenta as conclusões do trabalho realizado, ressaltando as contribuições e trabalhos correlatos e futuros.

O Apêndice A apresenta o Guia de Projeto Navegacional, onde são definidas regras para a criação do artefato Modelo Navegacional.

O Apêndice B apresenta, por completo, a extensão de UML WAE (*Web Application Extension*) proposta por J.Conallen [Conallen 99b].

O Apêndice C apresenta um resumo do framework de Análise e Projeto para aplicações Web proposto por A.Araújo [Araújo 01].

Capítulo 2

Engenharia de Software para Web

Nos dias de hoje, a Web está sendo utilizada pelas empresas na implementação de processos de negócio para seus funcionários, comunicação com seus parceiros, conexão e integração de seus sistemas, e efetuação de transações. Diante deste panorama, uma nova disciplina chamada de Engenharia de Software para Web está sendo desenvolvida, a qual trata do estabelecimento e uso de abordagens científicas, de engenharia, e gerenciamento para dar suporte ao processo de desenvolvimento de aplicações Web [Weippl 00].

Neste capítulo são descritas as aplicações Web e são feitas comparações com as aplicações tradicionais. Além disso, é apresentada a Engenharia de Software para Web, cujo propósito é estabelecer e usar princípios de engenharia e gerenciamento para o desenvolvimento de aplicações baseadas na Web.

2.1 Visão Geral

Da perspectiva da Engenharia de Software a Web é um novo domínio de aplicação, devido às características específicas das aplicações deste ambiente [Vasudevan 96], que as diferenciam das demais.

Diante deste panorama, neste capítulo são descritas as características específicas de aplicações Web e a necessidade de uma metodologia para orientar o processo de desenvolvimento deste tipo de aplicação, bem como o potencial que a Web tem para fornecer a infra-estrutura necessária à evolução de um produto de software.

2.2 Considerações sobre a Web

A Internet é uma rede de redes, através da qual programas e informações são intercambiados, quaisquer que sejam suas plataformas ou protocolos [Beveridge 98]. O principal serviço para busca de informação na Internet é o *World Wide Web* ou simplesmente Web. A Web é baseada no paradigma de hipertexto [Rossi 99], entretanto, foi introduzida uma tecnologia mais abrangente, chamada hipermídia, que manipula textos e multimídia de uma forma mais poderosa e intuitiva.

Segundo [Hanneghan 96], entre os aspectos que fazem a Web importante estão: sistema de hipertexto em rede, arquitetura distribuída sobre a qual a portabilidade de aplicações pode ser baseada, crescimento global exponencial, e rede aberta que pode ser acessada com custo relativamente baixo por grandes e pequenas empresas.

Além disso, destacam-se os seguintes motivos pelos quais as organizações estão adotando a Web [Hanneghan 96]: meio de comunicação interno e externo, fácil integração com logísticas corporativas, obtenção e manutenção de vantagem competitiva, meio para colaboração e desenvolvimento, recuperação e utilização de informação, marketing e vendas, transmissão de dados, e criação de uma presença corporativa.

A popularidade da Web, como infra-estrutura de suporte a aplicações, deve-se a vários fatores, entre os quais estão: a redução de custo de treinamento de pessoal, já que muitas pessoas estão habituadas a utilizar um *browser* (navegador) Web e correio eletrônico; e a comodidade de clientes/usuários efetuarem transações, através de acesso remoto.

2.3 Aplicações Web

Uma aplicação Web é um sistema que usa a Internet como meio de comunicação primário entre os usuários e o sistema. A distinção entre aplicações Web e *Web sites* é sutil, e consiste em: a aplicação que permite o usuário efetuar uma transação de negócio no servidor é considerada uma aplicação Web, caso contrário é um *Web site*.

Desenvolver uma aplicação Web não é somente uma questão tecnológica, mas também uma questão organizacional, política e cultural. O desenvolvimento destas aplicações envolve uma mudança na estruturação e apresentação da informação [Araújo 01].

Aplicações Web podem ser disponibilizadas para uma grande quantidade de usuários dentro de uma mesma organização (Intranet), através de várias organizações (Extranet) e sobre a Internet, diferentemente de aplicações tradicionais.

As tendências indicam alto interesse da maioria das organizações em desenvolver ou portar seus sistemas para a plataforma Web. Há três razões principais para isto: alta popularidade da tecnologia entre clientes em potencial, baixo custo de investimento, e alta possibilidade de alcance dos objetivos pretendidos. Em muitas organizações, uma das principais funções dos CIOs (*Chief Information Officers*) é explorar a Web à procura de benefícios comerciais e oportunidades de negócios [Hech 98].

Apesar da grande penetração da Web em uma variedade de mercados, está se tornando claro que a estrutura atual não está bem projetada para o acesso customizado à informação. Para tanto, projetista de aplicações Web têm de resolver questões de autoria, navegação, organização, gerenciamento e entrega de uma grande quantidade de informação via Web [Araújo 01].

2.4 Aplicações Web *versus* Aplicações Tradicionais

As principais diferenças entre aplicações Web e aplicações tradicionais referem-se a questões sobre navegação, organização da interface e implementação [Rossi 99]. Nos parágrafos a seguir, algumas dessas questões são apresentadas.

Referentes à Navegação

- O que constitui uma unidade de informação com respeito à navegação?
- Como estabelecer o que são links significantes entre unidades de informação?
- Onde o usuário inicia a navegação?
- Como organizar o espaço de navegação, isto é, estabelecer as seqüências possíveis de unidades de informação, através das quais o usuário irá navegar?
- Se uma interface Web está sendo adicionada para um sistema existente, como mapear os objetos de dados existentes sobre unidades de informação, e quais relacionamentos no domínio do problema devem ser mapeados em links?

Referente à Organização da Interface

- Quais objetos de interface o usuário irá perceber? Como relacionar estes objetos aos objetos de navegação?
- Como a interface se comportará quando usada pelo usuário?
- Como operações de navegação serão distinguidas das operações de interface e de processamento de dados?
- Como o usuário estará hábil a perceber sua localização no espaço de navegação?

Referente à Implementação

- Como as unidades de informação são mapeadas sobre páginas?
- Como operações de navegação são implementadas?
- Como outros objetos de interface são implementados?
- Como base de dados existentes são integradas à aplicação?

As aplicações Web provavelmente se tornarão mais universais do que as aplicações cliente/servidor se tornaram a uma década atrás, com um impacto exponencialmente maior em nossas vidas, simplesmente porque a Web tem potencialmente uma audiência muito mais ampla do que as aplicações cliente/servidor baseadas em redes proprietárias.

Apesar das diferenças, muitos dos problemas relacionados a aplicações Web seguem estratégias de aplicações tradicionais [Clemons 91]. Desta forma, muitas das soluções relacionadas a aplicações tradicionais também são aplicadas a aplicações Web.

2.5 A Necessidade de uma Metodologia

O crescimento da Web tem tido um impacto significativo sobre os diversos setores de negócios, bem como nas áreas de educação, governo, entretenimento e sobre nossa vida pessoal e profissional. As empresas estão desenvolvendo ou migrando seus sistemas corporativos para a plataforma Web, devido em grande parte à globalização da economia, tendo em vista que a área de atuação de uma empresa pode ser em âmbito mundial.

Entretanto, na maioria dos casos, as técnicas de desenvolvimento usadas para produzir aplicações Web têm sido *ad hoc*. Dificilmente, qualquer atenção é dada para métodos e processos sistemáticos de desenvolvimento, técnicas de medição e avaliação de qualidade e para o planejamento e gerenciamento de projetos. A maior parte dos esforços de desenvolvimento de aplicações Web atuais e das práticas de gerenciamento têm se baseado no conhecimento e experiência dos desenvolvedores e de suas próprias práticas de trabalho.

Na ausência de um processo disciplinado para desenvolvimento de aplicações Web, sérios problemas podem acontecer no processo de desenvolvimento, implantação, operação e manutenção destas aplicações. Aplicações desenvolvidas sem técnicas mais rigorosas têm uma alta probabilidade de falhar. Caso ocorra uma falha muito grave, a confiança na Web pode ser irreparavelmente abalada, causando uma grave crise [Zelnick 98]. Uma potencial crise na Web seria mais séria e difundida do que a tradicional crise de software [Gibbs 94].

Para evitar uma possível crise na Web e alcançar maior sucesso no desenvolvimento de aplicações Web complexas, há uma pressão por técnicas disciplinadas, novos métodos e ferramentas para desenvolvimento, implantação e avaliação destas aplicações. Tais técnicas devem levar em consideração as características especiais da nova mídia, os ambientes operacionais, os cenários e multiplicidade de perfis de usuários e o tipo, experiência e conhecimento das pessoas envolvidas no processo de construção das aplicações. Em [Koch 99] é feito um estudo comparativo entre vários métodos de desenvolvimento de aplicações hipermídia, entre as quais as aplicações Web podem ser inseridas.

A Engenharia de Software para Web deve se preocupar com o estabelecimento e uso de princípios de engenharia, gerenciamento e de técnicas sistemáticas e disciplinadas, de modo a alcançar aplicações Web de alta qualidade. Além disso, devem ser incorporados alguns dos princípios bem sucedidos da engenharia de software tradicional, adaptando-os para a natureza mais aberta e flexível da Web e de suas aplicações.

Desta forma, o núcleo de Engenharia de Software para Web [Hansen 99] deve incluir: especificação e análise de requisitos; metodologias e técnicas de desenvolvimento; integração com sistemas legados; migração de sistemas legados para ambientes Web; desenvolvimento de aplicações de tempo real para a Web; teste, verificação, avaliação, controle e garantia de qualidade; configuração e gerenciamento de projetos; métricas para estimativa dos esforços de desenvolvimento; especificação e avaliação de performance; atualização e manutenção; aspectos humanos e culturais; desenvolvimento centrado no usuário, incluindo modelagem, envolvimento e *feedback* para/do o usuário; educação e treinamento.

2.6 Adoção à Tecnologia Web

Apesar da enorme difusão e euforia associada à utilização da tecnologia Web, tem se tornado claro que esta tecnologia não é tão fácil de usar quanto muitas organizações supunham. Muitas organizações ainda têm encontrado dificuldade em resolver algumas questões básicas relacionadas à adoção da tecnologia Web [Vasudevan 96], tais como, seleção de tecnologias e padrões apropriados, planejamento de orçamento de projetos para a Web e treinamento de pessoal nestas novas tecnologias.

A emergência de novas tecnologias, como Web *Data Warehousing*, levantam questões de que não é somente necessária a intenção em se adotar uma tecnologia, precisa-se estar capacitado para tal. Cada potencial usuário tem que criar o conhecimento necessário para se beneficiar do uso desta tecnologia em seu contexto de negócios.

Em geral, três níveis de tecnologia Web são identificados [Nambisan 99]: acesso à informação (nível 1), trabalho colaborativo (nível 2) e transações de negócio (nível 3).

- Nível 1 – Acesso à Informação: este é o nível mais baixo, onde a tecnologia é usada como ferramenta para disseminação de informação sobre produtos e serviços, políticas organizacionais, missão corporativa e para partes internas e externas (clientes, empregados e investidores). Exemplos deste nível são Web *sites* corporativos e Intranets.

- Nível 2 – Trabalho Colaborativo: este é o nível intermediário, onde a tecnologia facilita a comunicação interna e externa, compartilhamento e localização de documentos em tempo real. Exemplos deste nível são Intranets e Extranets corporativas, e videoconferência.

- Nível 3 – Transações de Negócios: este é o nível mais alto, onde a tecnologia fornece suporte direto para transações e processos de negócios dentro e fora da organização. Exemplos deste nível são comércio eletrônico e ERP (*Enterprise Resource Planning*) estendido para Internet.

Estudos recentes indicam que a maior parte das organizações ainda está usando a Web primariamente para a disseminação de informação, ou seja, adotando tecnologia no nível 1. A adoção de tecnologia nos níveis mais altos pode ser considerada uma barreira de conhecimento. Estas barreiras de conhecimento podem ser classificadas em três categorias, relacionadas à tecnologia, ao projeto e à aplicação [Nambisan 99].

- Relacionadas à Tecnologia: falta de infra-estrutura apropriada de software e hardware, características tecnológicas, segurança e padrões. Estas barreiras são intensificadas pelo fato de que muitas das tecnologias Web não são maduras, fazendo com que a escolha de uma alternativa tecnológica seja um verdadeiro desafio.

- Relacionadas ao Projeto: falta de conhecimento relativo a recursos financeiros e humanos, duração do processo de desenvolvimento e liderança do projeto.

- Relacionadas à Aplicação: falta de conhecimento relativo aos objetivos de negócios específicos que serão implementados pela aplicação, o potencial de integração da aplicação com aplicações de TI existentes, e o impacto da aplicação Web nos sistemas e estrutura atuais da organização.

A Figura 2-1 apresenta a relação entre as barreiras de conhecimento e os níveis de adoção de tecnologia Web.

Barreira de Conhecimento	Nível 1	Nível 2	Nível 3
Relacionada à Tecnologia	- Seleção de tecnologias e padrões apropriados. - Introdução de flexibilidade técnica para acomodar futuros planos de integração.	- Obrigação de padrões tecnológicos através de unidades.	- Adaptação a mudanças na tecnologia sem impacto nos negócios. - Garantia da conformidade com regulamentos
Relacionada ao Projeto	- Identificação dos requisitos do projeto. - Resolução de questões de liderança de projeto.	- Seleção de metodologias de desenvolvimento apropriadas. - Resolução dos custos do projeto Web.	- Compartilhamento do custo/lucro para projetos Web com outros times de aplicações de IS.
Relacionada à Aplicação	- Elaboração de políticas de propriedade de dados corporativos. - Compartilhamento de informação/políticas de regulação de conteúdo. - Avaliação de requisitos de segurança.	- Estabelecimento de ligações entre negócios e aplicações Web. - Avaliação do impacto sobre a estrutura organizacional e sistemas.	- Reengenharia de processos de negócio para explorar a tecnologia Web. - Redefinição de relacionamentos intra-organizacionais.

Figura 2-1 Barreiras de conhecimento na adoção de tecnologias Web

2.6.1 Exemplos de Tecnologias de Implementação

Para serem dinâmicas e com interoperabilidade total com banco de dados, as aplicações Web utilizam tecnologias de implementação específicas. A seguir, algumas dessas tecnologias são apresentadas.

- Java: linguagem de programação de propósito geral, dinâmica, *multi-thread*, independente de plataforma e orientada a objeto [Sun 01]. Estas características a tornam uma das linguagens de programação que melhor se adaptam ao desenvolvimento de aplicações Web.

- *Applets*: programas Java que incluem funcionalidades específicas em páginas Web, como animação.

- *Servlets*: programas Java usados nos servidores Web para processar solicitações recebidas dos *browsers* clientes.

- *Java Server Pages* (JSP): usadas para encapsular a apresentação dos dados processados pelos *servlets*. O objetivo é separar a lógica do programa que é implementada como um *servlet* da apresentação feita pela JSP.

- *Java Script*: linguagem baseada em scripts, utilizada em aplicações Web para validar informações no cliente antes do processamento no servidor.

- *Java Beans*: componente de software reusável, tais como botões, que pode ser manipulado em uma ferramenta de desenvolvimento.

- *Enterprise Java Beans* (EJBs): objeto remoto e não visual, independente de plataforma, projetado para executar no servidor e ser invocado pelos clientes.

- *Active Server Page* (ASP): aplicação para construção de páginas dinâmicas e interativas.

- *Common Gateway Interface Script* (script CGI): programa que pode ser executado em um servidor Web para responder a uma requisição.

- *Common Object Request Broker Architecture* (CORBA): padrão que fornece uma variedade de serviços que capacitam componentes reusáveis a se comunicarem com outros componentes, independente de localização e/ou plataforma.

- *Java Database Connectivy* (JDBC): especificação de conectividade com banco de dados para a linguagem Java.

2.7 Desenvolvimento de Software através da Web

A Web tem potencial para fornecer a infra-estrutura necessária a um ambiente de engenharia de software global que suporte a evolução de um produto de software, independente de sua localização e número de pessoas envolvidas. Este ambiente pode dar suporte a inclusão, exclusão, e migração de participantes sem a necessidade de mudanças na infra-estrutura, limitando o efeito destas alterações no projeto como um todo.

A Web como ambiente de desenvolvimento possibilita que notas, diagramas e croquis estejam disponíveis a qualquer um e acessíveis a qualquer hora. Todo o material de projeto pode ser examinado, procurado e editado on-line; outros benefícios incluem suporte para controle de versões, estabelecimento de relacionamento hipermídia entre produtos de trabalho, pessoas e tarefas, e capacidade de rastreamento de dependências relacionadas. Planos, projetos, especificações e código, de projetos anteriores, podem ser localizados e reusados, parcial ou completamente. Os usuários também podem participar diretamente do desenvolvimento, sendo limitados unicamente pelas restrições de acesso determinadas pela organização de desenvolvimento. Como exemplo, pode-se citar o *Apache Hypertext Transfer Protocol* [Fielding 97], que foi desenvolvido por vários pesquisadores distribuídos pelo mundo, através da colaboração via ambiente Web.

A Web pode também capacitar produtos e seus processos associados a serem dispersos globalmente, enquanto permanecem altamente conectados e dinâmicos. Produtos de software podem ser mais bem projetados e construídos, desde que times de especialistas (projetistas, analistas, programadores, testadores e outros) dispersos globalmente possam ser agrupados no ciberespaço.

Uma estrutura de comunicação e interação baseada na Web torna possível também que produtos possam permanecer ligados a seu ambiente de desenvolvimento, aumentando a qualidade de suporte da organização. Tais ligações ou conexões podem dar suporte otimizações baseadas em padrões de observações de uso, atualizações on-line e contínua avaliação de qualidade. As etapas de treinamento de usuários, manutenção e evolução podem ser facilitadas por links para o processo suportado no ambiente de desenvolvimento. O reuso de componentes pode também ser incrementado através de uma conveniente biblioteca de componentes de software de alcance global.

Apesar de ser importante ressaltar a característica da Web de fornecer suporte e infra-estrutura ao desenvolvimento de software, neste trabalho esta característica não será abordada. Consideramos a utilização desta característica da Web como uma futura evolução deste trabalho.

2.8 Considerações Finais

Neste capítulo foram apresentadas as características específicas de aplicações Web que as diferenciam das aplicações tradicionais, bem como a Engenharia de Software para Web cujo propósito é a utilização de princípios de engenharia e gerenciamento, e de técnicas sistemáticas e disciplinadas para o desenvolvimento, implantação e manutenção de aplicações Web de alta qualidade. Dentro deste contexto, o próximo capítulo apresenta extensões de UML para aplicações Web, que são mecanismos utilizados para modelagem das características específicas deste tipo de aplicação.

Capítulo 3

Extensões de UML para Aplicações Web

UML – *Unified Modeling Language*, é uma linguagem padrão para modelagem de sistemas, principalmente os orientados a objetos, é amplamente utilizada no meio comercial e no meio acadêmico, e existem inúmeras ferramentas CASE que a suportam. Entretanto, por ser genérica, não satisfaz as necessidades de todos os tipos de aplicação, sendo então necessária uma extensão da linguagem para atender situações específicas, como o desenvolvimento de aplicações Web.

Uma extensão de UML é uma maneira ordenada de adicionar nova semântica para a notação de UML. O mecanismo de extensão permite a inclusão de: novos atributos, diferentes semânticas e restrições adicionais a elementos da modelagem, de forma a atender mais satisfatoriamente aspectos específicos de um tipo de aplicação.

Neste capítulo, é apresentado o estado da arte das extensões de UML para aplicações Web. São apresentadas abordagens propostas por J.Conallen [Conallen 99b], N.Koch [Koch 01] e L.Baresi [Baresi 01]. No final do capítulo, é apresentada uma comparação entre essas abordagens.

3.1 Visão Geral

A UML é a linguagem padrão para modelagem dos mais diferentes tipos de aplicação. Devido a esta generecidade, os projetistas da UML reconhecem que a linguagem nem sempre é perfeita (ideal) para todas as situações. Há vezes onde o processo de desenvolvimento é mais bem servido se informações adicionais forem capturadas, ou se diferentes semânticas forem aplicadas a certos elementos da modelagem. Para tanto, a UML pode ser estendida de forma a adicionar novos elementos à sua notação, através da criação de estereótipos, *tagged values* e restrições.

Extensões de UML foram criadas para melhor atender a modelagem de aplicações Web e, desta forma, satisfazer aspectos específicos deste tipo de aplicação, como a navegação. Neste capítulo são apresentadas algumas extensões de UML para aplicações Web.

3.2 Extensões de UML

Uma extensão para UML é expressa em termos de estereótipos, *tagged values*, e restrições (*constraints*). Combinados, esses mecanismos possibilitam criar novos tipos de blocos de construção que podem ser usados nos modelos.

- Estereótipo: é uma extensão do vocabulário da linguagem. Um estereótipo permite anexar um novo significado semântico a um elemento do modelo. Estereótipos podem ser aplicados em todos os elementos do modelo e são expressos como uma string entre << >> ou como um novo ícone.

- *Tagged Value*: é uma extensão para uma propriedade de um elemento do modelo. Muitos elementos do modelo têm propriedades associadas a eles. Classes, por exemplo, tem nome, visibilidade, persistência, entre outros atributos associados. Um *tagged value* é a definição de uma nova propriedade que pode ser associada com um elemento do modelo, sendo representado como uma string entre colchetes [ ].

- Restrição: é uma extensão da semântica da linguagem. Uma restrição especifica as condições sobre as quais o modelo pode ser considerado “bem formado”, sendo expressa como uma string entre chaves { }.

Nas seções a seguir, são apresentadas extensões de UML para aplicações Web.

3.3 WAE – *Web Application Extension*

Nesta seção, é apresentada a WAE (*Web Application Extension*), proposta por J.Conallen [Conallen 99a,Conallen 99b], para auxiliar a modelagem de aplicações Web. Esta extensão de UML identifica os elementos específicos para construção de uma aplicação Web, e está detalhada no apêndice B. Nos parágrafos a seguir, são apresentados os principais elementos desta extensão de UML.

Página Web

A página Web (página) necessita ser modelada, pois é um dos artefatos primários de uma aplicação Web. Uma página é representada no modelo através de duas classes distintas estereotipadas por <<server page>> e <<client page>>. Qualquer página em uma aplicação Web que tem funcionalidade tanto no servidor como no cliente pode ser representada no modelo como duas classes separadas, mesmo que suas implementações estejam em um mesmo arquivo ou componente.

Deste modo, métodos servidores e variáveis do escopo da página são contidos em uma classe no modelo, estereotipada por <<server page>>. Os métodos da classe representam os scripts do lado servidor, sub rotinas e funções. Os scripts do lado cliente e formatação da interface gráfica não são parte do escopo da página servidor. Uma página servidor pode ter relacionamentos com componentes existentes no servidor, tais como objetos de negócio nas camadas do sistema ou componentes de acesso a dados.

As páginas cliente são similarmente representadas no modelo como classes estereotipadas por <<client page>>. Os atributos da página cliente são variáveis do escopo da página e funções que executam no *browser* cliente. As páginas cliente são associadas com componentes que executam no cliente, como *Java Applets* e controles *ActiveX*.

Há um relacionamento fundamental entre os estereótipos cliente e servidor de uma página Web. Uma página servidor geralmente constrói (monta) o resultado em uma página cliente. Este relacionamento é representado no modelo através de uma associação estereotipada por <<build>>. A Figura 3-1 apresenta um exemplo de relacionamento entre páginas Web, que indica qual página servidor é responsável pela construção de uma determinada página cliente.

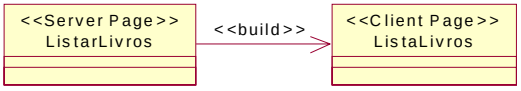


Figura 3-1 Construção de uma página Cliente

Uma outra facilidade de algumas tecnologias de desenvolvimento de aplicações Web é a habilidade de redirecionar a requisição de processamento para uma outra página servidor. Este relacionamento é representado no modelo através de uma associação estereotipada por <<redirect>>. O redirecionamento é uma característica muito útil para o reuso em aplicações Web. A Figura 3-2 apresenta um exemplo de redirecionamento de processamento entre páginas servidor.

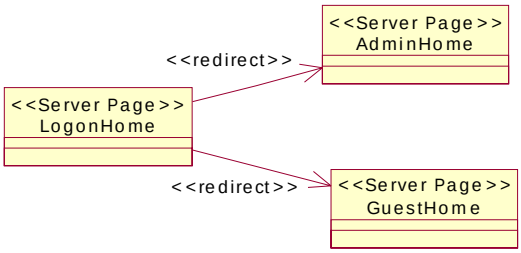


Figura 3-2 Redirecionamento de processamento para uma página Servidor

Um relacionamento adicional que é importante em projetos de aplicações Web é o hiper link (link). As páginas cliente freqüentemente contêm links para outras páginas Web (páginas cliente e/ou servidor) Este relacionamento é representado no modelo através de uma associação estereotipada por <<link>>. A Figura 3-3 apresenta um exemplo de relacionamentos entre páginas Web, que indica quais páginas Web são referenciadas através de link por uma página cliente.

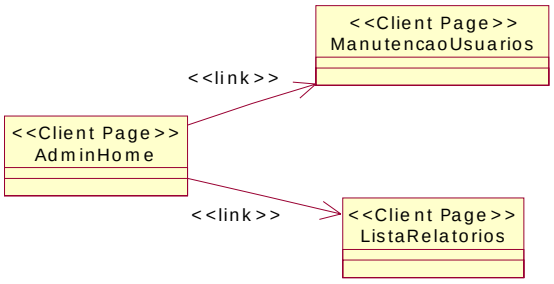


Figura 3-3 Link entre páginas Web

Formulário (Form)

São definidos estereótipos adicionais para o uso de formulários HTML em aplicações Web. Os formulários contêm atributos adicionais que podem não ser apropriados no contexto da página cliente inteira e é possível ter múltiplos formulários em uma única página cliente, cada um apontando para uma ação diferente. Desta forma, os formulários são representados no modelo através de uma classe estereotipada por <<form>>.

Uma classe estereotipada por <<form>> tem como atributos os campos do formulário. Estes atributos são estereotipados para representar o tipo dos campos (entrada, lista, botão, seleção, etc) no formulário. A contenção é a relação formal entre uma página cliente e um formulário, ou seja, páginas cliente contêm formulários. Isto é representado no modelo através da associação do tipo agregação entre a classe que representa a página cliente e a classe que representa o formulário.

Um relacionamento é necessário para identificar qual página Web processa os dados submetidos por um formulário. Este relacionamento é representado no modelo através de uma associação estereotipada por <<submit>>. A Figura 3-4 apresenta um exemplo da utilização de formulários em aplicações Web.

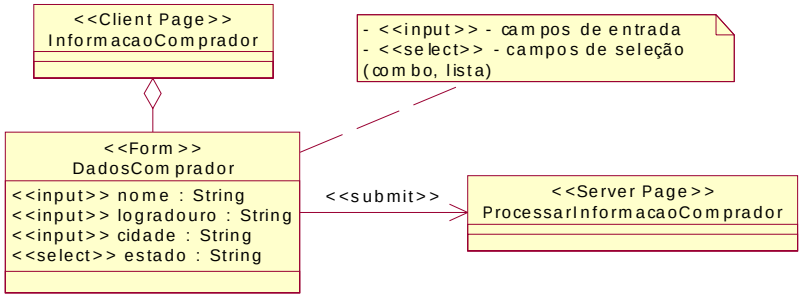


Figura 3-4 Uso de Formulários em aplicações Web

Frame

Uma interface adicional (e elemento de projeto) disponível em aplicações Web é o frame. O frame permite a representação de múltiplas páginas Web ao mesmo tempo, e permite ao projetista Web dividir uma janela do *browser* em subáreas retangulares (*panes*) que exibem diferentes páginas Web.

Os frames são implementados em HTML pela definição de um *frameset*. Um *frameset* é um tipo especial de página Web que divide sua área de visualização em múltiplas subáreas, com cada uma exibindo sua própria página Web. Isto é representado no modelo através de uma classe estereotipada por <<frameset>>.

Cada uma das subáreas de visualização em uma página Web é um *target*. Um *target* no *frameset* é um frame nomeado que outras páginas cliente podem requisitar através de páginas Web. Isto é representado no modelo através de uma classe estereotipada por <<target>>. O relacionamento entre páginas cliente e um *target* é representado no modelo através de uma associação estereotipada por <<targeted link>>. A Figura 3-5 apresenta um exemplo do uso de frames em aplicações Web.

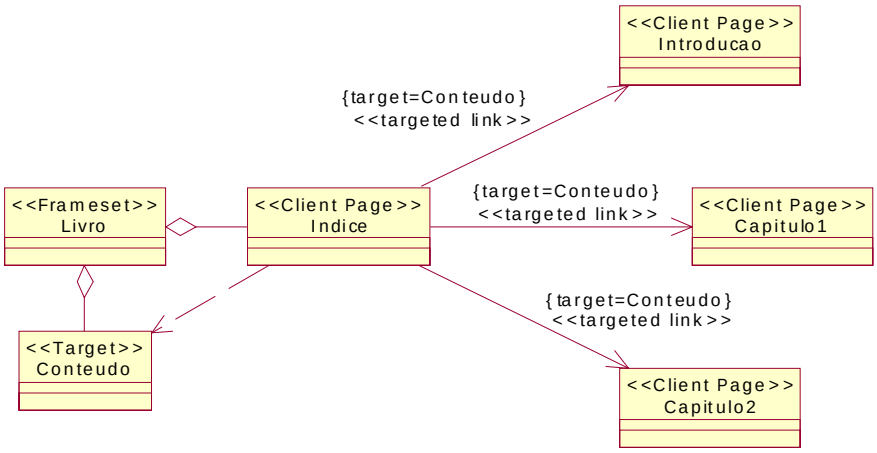


Figura 3-5 Uso de Frames em aplicações Web

3.4 Extensão de UML para Modelagem Navegacional

O processo de construção *Ad hoc* para aplicações Web foca no conteúdo e na apresentação, com pequena atenção a estrutura navegacional. A qualidade de uma aplicação Web depende não somente da riqueza do conteúdo e do projeto gráfico de alta qualidade, mas também de uma navegação bem estruturada.

Nesta seção, é apresentada uma extensão de UML, proposta por N.Koch [Koch 01], para modelagem da navegação e da apresentação de aplicações Web. Esta extensão de UML é parte de uma metodologia para análise e projeto de aplicações Web [Baumeister 99,Hennicker 00], baseada no OOHDM – *Object Oriented Hypermedia Design Method* [Rossi 96,Schwabe 96]. Esta metodologia trata separadamente o conteúdo, a navegação e a apresentação da aplicação, através da execução dos seguintes passos: Projeto Conceitual, Projeto Navegacional e Projeto da Apresentação.

Projeto Conceitual

O objetivo deste passo é capturar a semântica do domínio do problema e, desta forma, produzir o Modelo Conceitual da aplicação, representado pelo diagrama de classes de UML. As atividades do Projeto Conceitual consistem em: encontrar classes, especificar atributos e operações, definir associações entre as classes, definir estruturas hierárquicas, encontrar dependências, identificar interfaces e definir restrições. Técnicas de modelagem como agregação, composição, generalização e especialização são usadas para obter esses propósitos. A Figura 3-6 apresenta o Modelo Conceitual de uma aplicação Web que controla os serviços prestados por uma empresa.

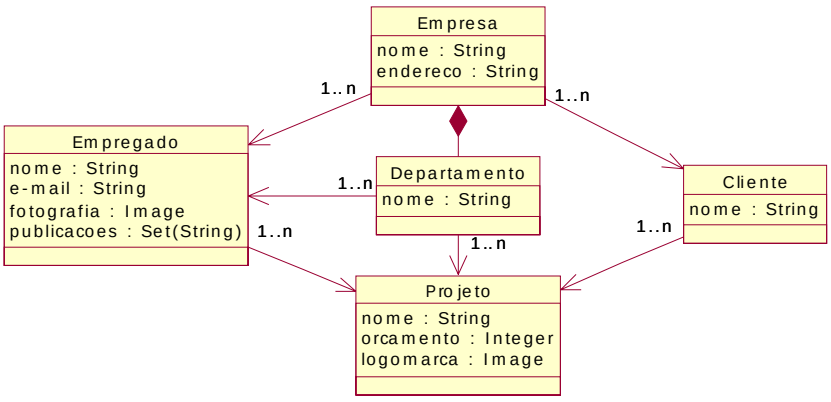


Figura 3-6 Modelo Conceitual da aplicação "Controlador de Serviços"

Projeto Navegacional

O objetivo deste passo é produzir o Modelo Navegacional da aplicação, que pode ser considerado uma outra visão do Modelo Conceitual. Para tanto, são necessários dois passos. No primeiro passo o Modelo do Espaço Navegacional é definido e, no segundo passo, o Modelo da Estrutura Navegacional é construído.

O Modelo do Espaço Navegacional define uma visão do Modelo Conceitual mostrando quais classes (chamadas classes navegacionais) podem ser visitadas através da navegação da aplicação. Este modelo é representado por um diagrama de classes de UML, e é construído com classes estereotipadas por <<navigational class>> e com associações estereotipadas por <<direct navigability>>. A Figura 3-7 apresenta o Modelo do Espaço Navegacional derivado do Modelo Conceitual apresentado na Figura 3-6.

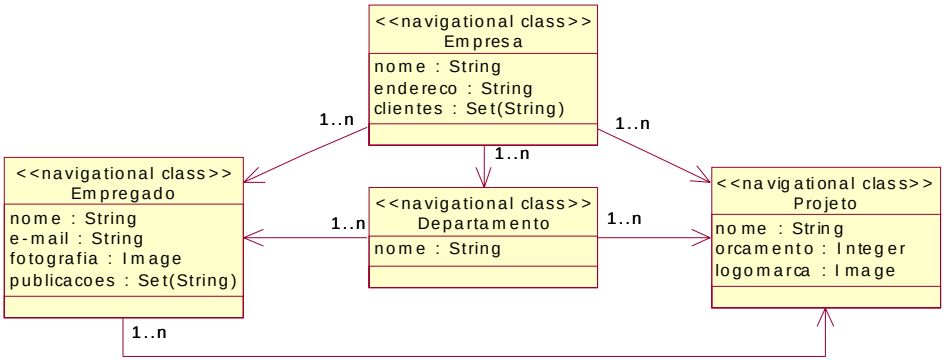


Figura 3-7 Modelo do Espaço Navegacional da aplicação "Controlador de Serviços"

Para construção do Modelo do Espaço Navegacional, o projetista de navegação deve orientar-se pelas seguintes regras [Hennicker 00]:

1. Classes do Modelo Conceitual que são relevantes para a navegação são incluídas como classes navegacionais no Modelo do Espaço Navegacional. Se uma classe conceitual não é visitada no Modelo de Casos de Uso, ela é irrelevante no processo navegacional e, portanto, é omitida no Modelo do Espaço Navegacional (tal como a classe Cliente do Modelo Conceitual apresentado na Figura 3-6).

2. Informações requeridas das classes conceituais omitidas podem ser mantidas como atributos de outras classes no Modelo do Espaço Navegacional (tal como o atributo *clientes* adicionado à classe navegacional Empresa). Todos os outros atributos das classes navegacionais são mapeados diretamente dos atributos da classe conceitual correspondente. Os atributos das classes conceituais tidos como irrelevantes para a apresentação são excluídos no Modelo do Espaço Navegacional.

3. Geralmente, novas associações são adicionadas para navegação direta evitando caminhos navegacionais de tamanho maior que um (tal como a associação introduzida entre as classes navegacionais Empresa e Projeto). Cenários descritos pelo Modelo de Casos de Uso fornecem a base para escolha de navegações diretas.

O Modelo da Estrutura Navegacional define a navegação da aplicação, isto é, como os objetos navegacionais (instâncias de classes navegacionais) são visitados. Este modelo é derivado do Modelo do Espaço Navegacional, e é representado por um diagrama de classes, que é construído utilizando elementos adicionais (estereótipos) para realizar navegação, tais como primitivas de acesso (menus, índices, *query* e *guided tour*) e contextos navegacionais. A Figura 3-8 apresenta o padrão de análise para estrutura navegacional [Koch 01].

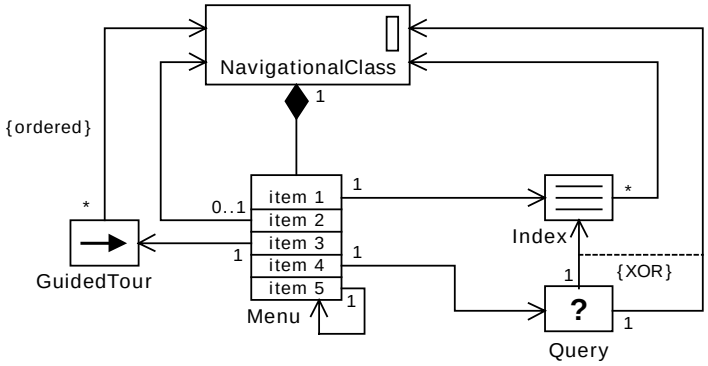


Figura 3-8 Padrão de Projeto para Estrutura Navegacional

A partir do Modelo do Espaço Navegacional, o projetista de navegação deve criar o Modelo da Estrutura Navegacional, seguindo o padrão de projeto apresentado na Figura 3-8. A Figura 3-9 apresenta o Modelo da Estrutura Navegacional da aplicação Web "Controlador de Serviços".

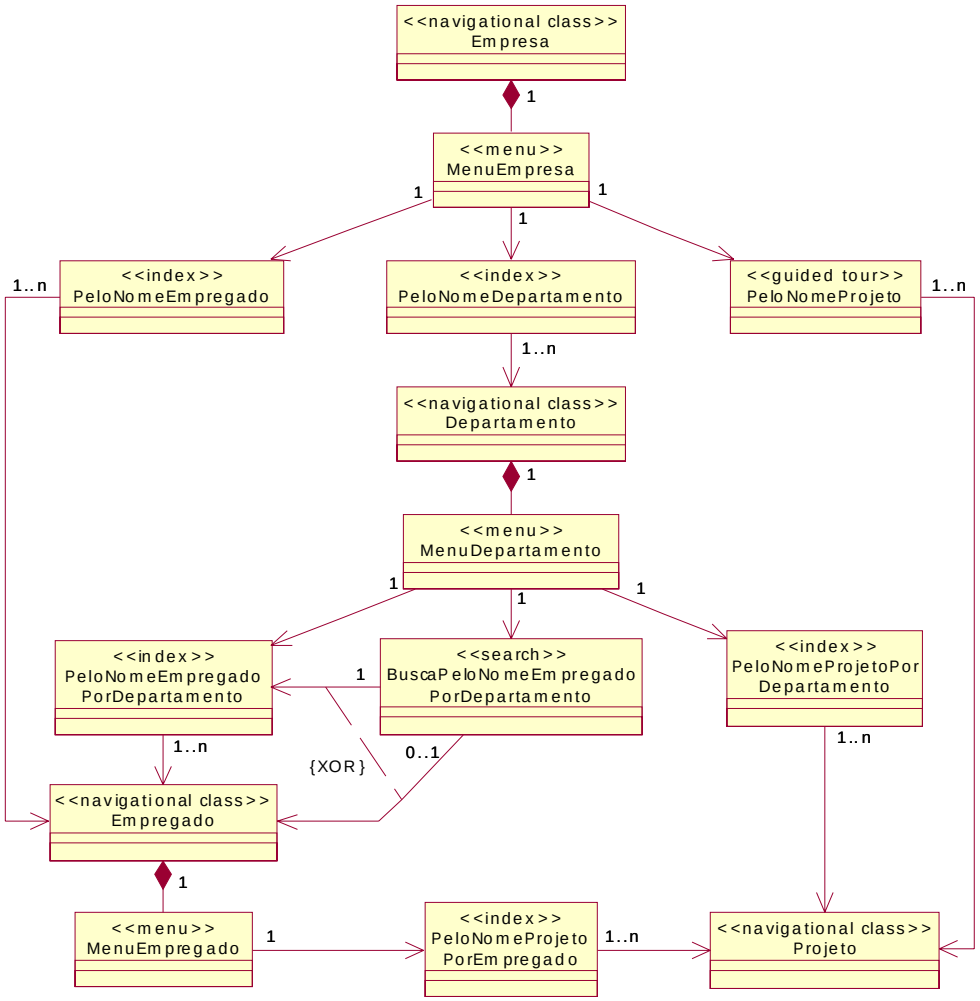


Figura 3-9 Modelo da Estrutura Navegacional da aplicação "Controlador de Serviços"

Projeto de Apresentação

O objetivo deste passo é modelar as interfaces gráficas da aplicação, mostrando como a estrutura navegacional é apresentada ao usuário. O Projeto de Apresentação define a aparência dos nós navegacionais e identifica quais objetos da interface gráfica ativam a navegação. Para tanto, são construídos um Modelo de Apresentação Estático e um Modelo de Apresentação Dinâmico.

O Modelo de Apresentação Estático é representado por diagramas de componentes de UML, que descrevem como as interfaces gráficas são construídas. As interfaces gráficas são definidas a partir de objetos de interface gráfica. Um objeto de interface gráfica pode ser um objeto primitivo como texto, imagem e botão, ou uma composição desses objetos primitivos. A Figura 3-10 apresenta partes do Modelo de Apresentação Estático da aplicação Web "Controlador de Serviços". O lado esquerdo da Figura 3-10 exibe a interface gráfica abstrata da classe navegacional Empresa, através do uso de *frameset*. O lado direito da Figura 3-10 exibe a organização dos campos que compõem a interface gráfica da classe navegacional Empregado.

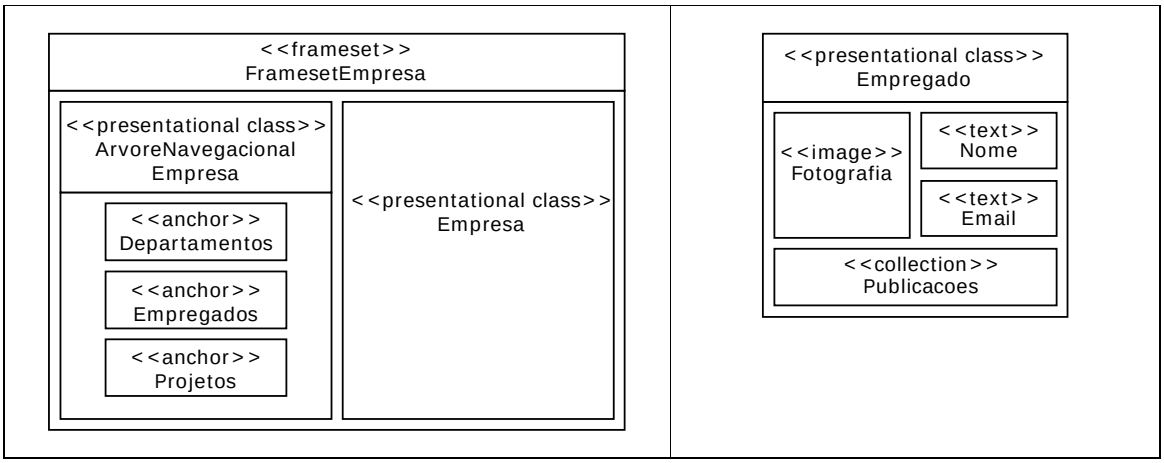


Figura 3-10 Partes do Modelo de Apresentação Estático da aplicação "Controlador de Serviços"

O Modelo de Apresentação Dinâmico é representado por diagramas de seqüência ou diagramas de estado de UML, que descrevem as colaborações e comportamentos dos objetos navegacionais e primitivas de acesso, ou seja, como a navegação é ativada. O exemplo deste modelo não é apresentado, pois o aspecto dinâmico das interfaces gráficas não é tratado neste trabalho.

3.4.1 Estereótipos para Modelagem Navegacional

A seguir, são detalhados os estereótipos desta extensão de UML, para modelagem da navegação e da apresentação de aplicações Web.

- Classe Navegacional (<<navigational class>>): representa a classe conceitual cujas instâncias são visitadas pelo usuário durante a navegação.

- Navegação Direta (<<direct navigability>>): representa a associação (navegação) de uma classe navegacional origem para uma classe navegacional destino.

- Índice (<<index>>): primitiva de acesso que representa uma coleção de objetos que possuem links para uma instância de uma classe navegacional.

- *Guided Tour* (<<guided tour>>): primitiva de acesso que representa um objeto que fornece acesso seqüencial a instâncias de uma classe navegacional.

- *Query* (<<query>>): representa um objeto que possui como atributo uma string que representa uma operação, como por exemplo, uma operação de seleção, cujo resultado pode ser mapeado em um link para uma instância de uma classe navegacional ou em um link para um índice.

- Menu (<<menu>>): primitiva de acesso que representa uma coleção de objetos que possuem links para uma instância de uma classe navegacional ou para uma outra primitiva de acesso.

- Contexto Navegacional (<<navigational context>>): representa uma seqüência ordenada de navegação entre instâncias de classes navegacionais.

- Classe de Apresentação (<<presentational class>>): representa a apresentação de uma classe navegacional ou de uma primitiva de acesso. Instâncias de classes de apresentação são *containers* de elementos de interface gráfica como texto (<<text>>), link (<<anchor>>), botão (<<button>>), imagem (<<image>>), áudio (<<audio>>), vídeo (<<video>>) e coleções (<<collection>>).

3.5 O Framework W2000

Nesta seção, é apresentado o framework W2000, proposto por L.Baresi [Baresi 01], para projeto de aplicações Web. Este framework é uma integração entre UML e HDM – *Hypermedia Design Model* [Garzotto 93], que consiste em: definir estereótipos e customização de diagramas para caracterizar HDM com UML, especificar guias para usar UML para especificar alguns dos aspectos dinâmicos e operacionais de aplicações Web, e refinar diagramas de Casos de Uso para descrever requisitos de alto nível, relacionados a aspectos funcionais e navegacionais.

O framework W2000 organiza as atividades do projeto em tarefas interdependentes, conforme apresentado na Figura 3-11. Cada atividade produz um modelo (representado por diagramas de UML), que descreve alguns aspectos da aplicação.

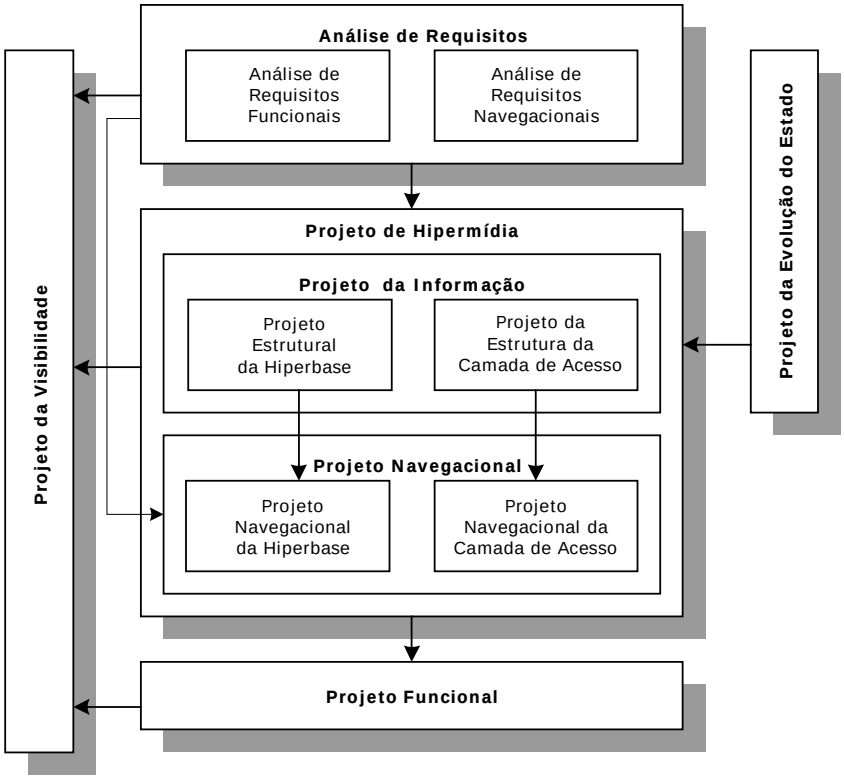


Figura 3-11 Atividades do framework W2000

Nos parágrafos a seguir, são detalhadas as atividades do framework W2000, apresentadas na Figura 3-11.

Análise de Requisitos

Esta atividade estende a análise de requisitos convencional para aplicações hipermídia. Ela consiste de duas sub-atividades: análise de requisitos funcionais e analise de requisitos navegacionais.

A Análise de Requisitos Funcionais identifica as principais funcionalidades da aplicação, e produz o Modelo de Casos de Uso Funcional, representado por um diagrama de Casos de Uso de UML.

A Análise de Requisitos Navegacionais destaca as estruturas navegacionais necessitadas pelos diferentes usuários da aplicação, e produz o Modelo de Casos de Uso Navegacional, representado por um diagrama de Casos de Uso de UML.

Projeto da Evolução do Estado

Esta atividade complementa a análise de requisitos e define como evoluir os conteúdos da aplicação, através de diagramas de estado de UML. Esta atividade não é obrigatória, mas é requerida para aplicações com comportamentos complexos.

Projeto de Hipermídia

Esta atividade especifica as estruturas de informação (Projeto da Informação) e os caminhos navegacionais (Projeto Navegacional) necessários para as várias classes de usuários.

O Projeto de Hipermídia é organizado em duas camadas distintas: a camada de hiperbase e a camada de acesso. A camada de hiperbase define os objetos de informação base, suas associações e os caminhos navegacionais através deles. A camada de acesso organiza os modos dos usuários iniciarem sua navegação dentro do espaço de informação.

O Projeto da Informação especifica e organiza os conteúdos da aplicação, através do Projeto Estrutural da Hiperbase e do Projeto da Estrutura da Camada de Acesso. Já o Projeto Navegacional, define como os usuários podem navegar nos elementos de informação e nas estruturas de acesso, através do Projeto Navegacional da Hiperbase e do Projeto Navegacional da Camada de Acesso. Todos os modelos produzidos são representados por diagramas de classes de UML, construídos com estereótipos apropriados.

Projeto Funcional

Esta atividade especifica as principais funcionalidades da aplicação e estende a especificação de funções padrões com algumas peculiaridades específicas para aplicações hipermídia. Os projetistas devem construir cenários para as principais funcionalidades, através de diagramas de interação de UML.

Projeto da Visibilidade

Diferentes usuários, em geral, têm uma perspectiva diferente da aplicação, com relação ao conteúdo e funcionalidades. O propósito desta atividade é especificar quais funcionalidades, estruturas de informação e caminhos navegacionais devem ser visíveis e para quem.

3.6 Considerações sobre as Extensões de UML

A WAE (*Web Application Extension*), proposta por J.Conallen [Conallen 99b], define um conjunto de estereótipos para modelagem de elementos específicos de aplicações Web, tais como, página cliente, página servidor, link, frame, formulário, entre outros. Entretanto, não atende a um aspecto bastante relevante em se tratando de aplicações Web, a navegação.

A extensão de UML proposta por N.Koch [Koch 01], abordagem orientada a objeto baseada no OOHDM, trata especificamente os aspectos de navegação e de apresentação de aplicações Web, porém, não define estereótipos para modelar elementos básicos, como páginas Web.

O framework W2000, proposto por L.Baresi [Baresi 01], é uma caracterização do HDM usando o meta-modelo da UML, trata os aspecto de navegação e de visibilidade, entretanto, não define um processo para tratar o aspecto de apresentação.

A Figura 3-12 apresenta uma síntese dos aspectos atendidos pelas extensões de UML.

	Elementos Específicos	Navegação	Apresentação
WAE	Atende		Atende
Koch (OOHDM + UML)		Atende	Atende
Framework W2000		Atende	

Figura 3-12 Aspectos atendidos pelas extensões de UML para aplicações Web

Após analisar os aspectos atendidos pelas extensões de UML para aplicações Web, conforme apresentado na Figura 3-12, chegamos as seguintes conclusões:

A abordagem proposta por J.Conallen (WAE) define estereótipos para modelagem dos elementos específicos de aplicações Web, tais como página cliente, página servidor e script. Além disso, define estereótipos para modelagem da apresentação das interfaces gráficas para o usuário.

A abordagem proposta por N.Koch utiliza a UML para modelar os aspectos de navegação e de apresentação de aplicações Web. Para tanto, utiliza o processo definido pela metodologia orientada a objetos para construção de sistemas hipermídia OOHDM.

O framework W2000, proposto por L.Baresi, utiliza a UML para caracterizar o HDM e, desta forma, atende aos aspectos de navegação e visibilidade de aplicações Web. Entretanto, o HDM é uma das mais antigas metodologias para construção de sistemas hipermídia e baseia-se na notação E-R (entidade e relacionamento), ou seja, não é orientada a objetos.

Diante deste panorama, a combinação entre as extensões de UML propostas por J.Conallen (WAE) e N.Koch (OOHDM + UML), mostram-se adequadas para tratar os aspectos específicos de aplicações Web orientadas a objetos.

3.7 Considerações Finais

Neste capítulo foram apresentadas extensões de UML para aplicações Web. Estas extensões estão mais detalhadas em [Souza 01]. Posteriormente, foram feitas considerações sobre quais aspectos são atendidos por cada uma destas extensões. Chegamos à conclusão que a combinação entre as abordagens propostas por J.Conallen (WAE) e N.Koch (OOHDM + UML) mostram-se adequadas para atender as características específicas de aplicações Web orientadas a objetos.

No próximo capítulo, é apresentado o RUP (*Rational Unified Process*) – metodologia genérica para o desenvolvimento de sistemas. Para o caso de aplicações Web esta metodologia deve ser adaptada para melhor atender o desenvolvimento deste tipo de aplicação. Parte desta adaptação consiste na criação de modelos que descrevam as características específicas das aplicações Web, utilizando para isso a combinação de extensões de UML eleitas neste capítulo.

Capítulo 4

O RUP - Rational Unified Process

Um processo é um conjunto de passos ordenados para alcançar um objetivo. Um processo de software é o conjunto de atividades e resultados associados que produzem um produto de software [Sommerville 96].

O RUP (*Rational Unified Process*) é um processo de software genérico para construção de sistemas orientados a objetos. Seu objetivo é garantir uma produção de software de alta qualidade que atenda as necessidades dos usuários dentro do prazo e do orçamento. O RUP utiliza as melhores práticas de desenvolvimento de software moderno, de forma que ele pode ser usado em uma grande faixa de projetos e organizações.

Neste capítulo, é apresentado o RUP (baseado na versão 2000.02.10), bem como suas características e fundamentos que orientam o processo de desenvolvimento de software.

4.1 Visão Geral

O RUP (*Rational Unified* Process) é um processo de software genérico que fornece uma abordagem disciplinada para assinalar tarefas e responsabilidades dentro do contexto do projeto, de modo a atender todo o ciclo de desenvolvimento de softwares de alta qualidade.

O RUP foi desenvolvido com o propósito de padronizar o processo de desenvolvimento de software. Para tanto, utiliza o melhor de várias técnicas de desenvolvimento de software e a UML – linguagem de modelagem padrão.

4.2 Conceitos Chaves do RUP

A seguir, são apresentados os conceitos chave utilizados para a explanação da metodologia RUP.

- Responsável – define o comportamento, as responsabilidades, e o papel desempenhado por um indivíduo ou uma equipe, dentro do contexto do projeto.

- Atividade – unidade de trabalho com um claro propósito, desempenhada por um responsável e que produz um resultado significante no contexto do projeto, geralmente expressa em termos de criação e atualização de artefatos.

- Artefato – peça de informação que é produzida, modificada ou usada pelo processo de desenvolvimento de software; pode ser um modelo, um elemento do modelo ou um documento.

- Fluxo – seqüência de atividades que produz um resultado de valor observável, representado no RUP através de um diagrama de atividade de UML, sendo que cada atividade deste diagrama corresponde a um subfluxo.

- Subfluxo – agrupamento de atividades, responsáveis envolvidos, artefatos de entrada e artefatos produzidos; utilizados para fornecer um alto nível de abstração e para estruturar um fluxo.

4.3 Características do RUP

O RUP é baseado em componentes, o que significa que a aplicação a ser construída será formada por componentes interconectados através de interfaces bem definidas. As principais características do RUP são: Dirigido a Casos de Uso; Centrado na Arquitetura; Iterativo e Incremental. Nas subseções a seguir, essas características são detalhadas.

4.3.1 Dirigido a Casos de Uso

Um sistema de software é construído para servir aos usuários. Conseqüentemente, para se construir um sistema de sucesso deve-se conhecer o que o usuário espera e necessita do sistema. Entende-se por usuário não somente pessoas, mas também outros sistemas interconectados.

Uma possível interação entre um usuário e o sistema pode ser representada por um Caso de Uso. Um Caso de Uso é um pedaço de funcionalidade do sistema que retorna ao usuário um resultado de valor observável. Os Casos de Uso representam os requisitos funcionais e juntos formam o Modelo de Casos de Uso que descreve todas as funcionalidades do sistema.

Os Casos de Uso não são somente uma ferramenta para especificação de requisitos. Eles também dirigem o projeto, a implementação e os testes, ou seja, dirigem o processo de desenvolvimento. Baseados no Modelo de Casos de Uso, projetistas criam uma série de modelos de projeto e implementação que realizam os Casos de Uso.

Dirigido a Casos de Uso significa que o processo de desenvolvimento segue um fluxo de ações para realização de Casos de Uso – isto procede através da execução dos fluxos do processo. Desta forma, os Casos de Uso são especificados, projetados e no fim, são as bases para os testadores construírem os casos de teste.

4.3.2 Centrado na Arquitetura

O conceito de arquitetura de software engloba os aspectos estáticos e dinâmicos mais significantes do sistema. A arquitetura é influenciada por muito fatores, tais como a plataforma de software, blocos de construção disponíveis para reuso, considerações de implantação, sistemas legados e requisitos não-funcionais. A arquitetura é uma visão do projeto como um todo, que torna visível as características mais importantes do mesmo.

O RUP fornece uma maneira metódica e sistemática para projetar, desenvolver e validar a arquitetura. São disponibilizados *templates* para descrições arquiteturais sobre visões arquiteturais[1], estilos arquiteturais, regras de projeto, e restrições.

Todo produto tem funções e forma. Estas duas forças devem ser balanceadas para que o produto possa ter sucesso. No caso de software, funções correspondem a Casos de Uso e a forma à arquitetura.

4.3.3 Iterativo e Incremental

O desenvolvimento de um produto de software comercial pode continuar por vários e vários anos. Por isso é prático que o trabalho seja dividido em pequenos ciclos ou mini-projetos. Cada mini-projeto é uma iteração que resulta em um incremento. Iterações referem-se a passos no fluxo de desenvolvimento, e incrementos a evoluções do produto. Sucessivas iterações constroem os artefatos de desenvolvimento a partir do estado que eles foram deixados ao final da iteração anterior.

Em toda iteração, o projetista identifica e especifica os Casos de Uso relevantes, cria um projeto usando a arquitetura selecionada com guia, implementa o projeto em componentes e verifica se os componentes satisfazem estes Casos de Uso. Se a iteração alcança suas metas, o desenvolvimento então, procede para a próxima iteração.

Os benefícios de um processo iterativo controlado são os seguintes: redução do risco de custos com despesas em um único incremento; redução do risco de atrasos no planejamento; aceleração do ritmo de todos os esforços de desenvolvimento porque os desenvolvedores trabalham mais efetivamente em direção a um resultado claro; reconhecimento de uma realidade freqüentemente ignorada – as necessidades do usuário e os requisitos correspondentes não podem ser completamente definidos de uma vez.

[1] Visão da arquitetura do sistema com foco na estrutura, modularidade, componentes essenciais, e os principias fluxos de controle.

4.4 Ciclo de Vida do Software no RUP

O RUP repete uma série de ciclos de evolução durante o processo de desenvolvimento de um sistema. Cada ciclo termina com uma versão do produto e consiste de quatro fases: Concepção, Elaboração, Construção e Transição. Cada fase é adicionalmente dividida em iterações, conforme apresentado na Figura 4-1.

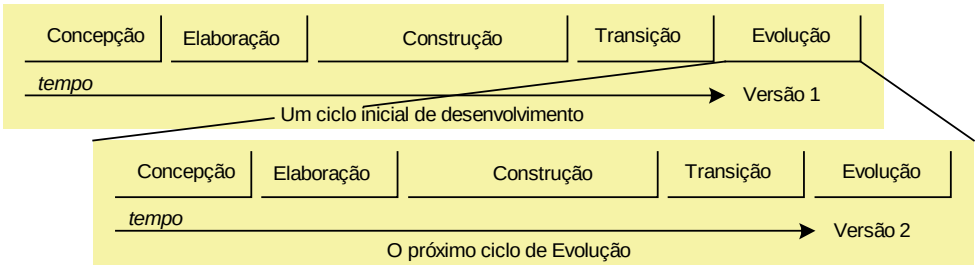


Figura 4-1 Ciclos e Fases do RUP

Fase de Concepção

Os objetivos desta fase são: estabelecimento do escopo do sistema e condições limites, incluindo uma visão operacional, o critério de aceitação e o que é para ser esperado do produto e o que não é; discriminação dos Casos de Uso críticos do sistema e restrições; exibição e/ou demonstração de pelo menos uma arquitetura candidata; estimação do custo total e prazo para o projeto inteiro; identificação dos potenciais riscos; preparação do ambiente de suporte para o projeto.

As atividades essenciais desta fase são as seguintes:

- Formular o escopo do projeto – capturar o contexto, os requisitos mais importantes e as restrições, para elaboração do critério de aceitação do produto final.
- Planejar e preparar o negócio – avaliar alternativas para gerenciamento de risco, planejar o projeto, e identificar restrições de custo e prazo.
- Sintetizar uma arquitetura candidata – avaliar alternativas entre fazer, comprar e reusar, de modo que custo, prazo e recursos possam ser estimados.
- Preparar o ambiente do projeto – selecionar ferramentas para o desenvolvimento.

Fase de Elaboração

Os objetivos desta fase são: garantir que a arquitetura e os requisitos estejam estáveis, e os riscos minimizados, para determinar custo e prazo para o restante do processo de desenvolvimento; controlar os riscos arquiteturais do projeto; estabelecimento de uma arquitetura e demonstração de que esta arquitetura suporta os requisitos do sistema com custo e tempo razoável; produção de um protótipo para identificação de deficiências de requisitos e projeto, reuso de componente, e demonstração para investidores, clientes, e usuários finais; estabelecimento do ambiente de suporte.

As atividades essenciais desta fase são as seguintes:

- Definir e validar a arquitetura.
- Refinar a Visão[2] – estabelecer um entendimento sólido dos Casos de Uso mais críticos que dirigem as decisões de planejamento e de arquitetura.
- Criar planos de iteração detalhados para a fase de Construção.
- Refinar a arquitetura e selecionar componentes – potenciais componentes são avaliados e as decisões de fazer/comprar/reusar são tomadas para determinar com confiança o custo e o prazo da fase de Construção.

Fase de Construção

Os objetivos desta fase são: minimizar custos de desenvolvimento, otimizando recursos e evitando re-trabalho; criação de versões usáveis do sistema (alfa, beta, e outras versões de teste); conclusão da análise, projeto, desenvolvimento e teste de todas as funcionalidades requeridas; desenvolvimento iterativo e incremental de um produto que esteja pronto para a transição para sua comunidade de usuários; decidir se o software e os usuários estão prontos para a aplicação ser implantada; obtenção de um certo grau de paralelismo no trabalho das equipes de desenvolvimento.

As atividades essenciais desta fase são: gerenciamento de recursos; complemento do desenvolvimento de componentes e testes; avaliação das versões do produto.

[2] Visão do cliente/usuário sobre o produto a ser desenvolvido.

Fase de Transição

Os objetivos desta fase são: beta testes para validar o sistema com relação às expectativas dos usuários; beta testes relativos à integração com sistemas legados; treinamento de usuários; obtenção da concordância dos *stakeholders*[3] de que o produto está consistente com o critério de aceitação.

As atividades essenciais desta fase são: execução dos planos de implantação; finalização do material de suporte ao usuário final; criação de versões do produto; obtenção de *feedback* do usuário; melhoramento do produto baseado no *feedback*; disponibilização dos produto para os usuários finais.

4.5 Fluxos do RUP

Basicamente, há dois tipos de fluxo no RUP: fluxos de Processo e fluxos de Suporte, conforme apresentado na Figura 4-2. Os fluxos de Processo correspondem às atividades de desenvolvimento: modelagem de negócios, requisitos, análise e projeto, implementação, testes e implantação. Os fluxos de Suporte correspondem às atividades de gerenciamento e infra-estrutura: gerenciamento de configuração e mudanças, gerenciamento de projeto e ambiente.

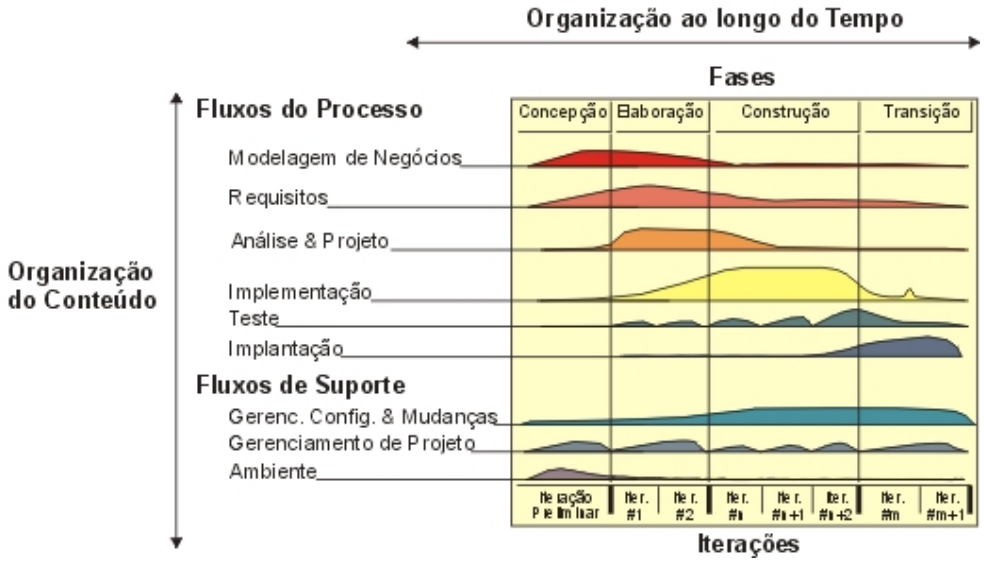


Figura 4-2 Fluxos do RUP

[3] Todas as pessoas relacionadas a um determinado projeto, desde usuários até desenvolvedores, clientes, gerentes, etc.

Nas subseções a seguir, são apresentados os fluxos do RUP. Os fluxos de Suporte são apresentados em menos detalhes, pois o foco desta dissertação está no processo de desenvolvimento.

4.5.1 Fluxo de Modelagem de Negócio

Os propósitos deste fluxo, apresentado na Figura 4-3, são os seguintes: entendimento da estrutura e da dinâmica da organização para a qual o sistema vai ser construído; entendimento dos problemas correntes da organização e identificação de potenciais melhorias; garantir que clientes, usuários finais, e desenvolvedores tenham um entendimento (visão) comum da organização; derivar os requisitos do sistema para dar suporte a organização.

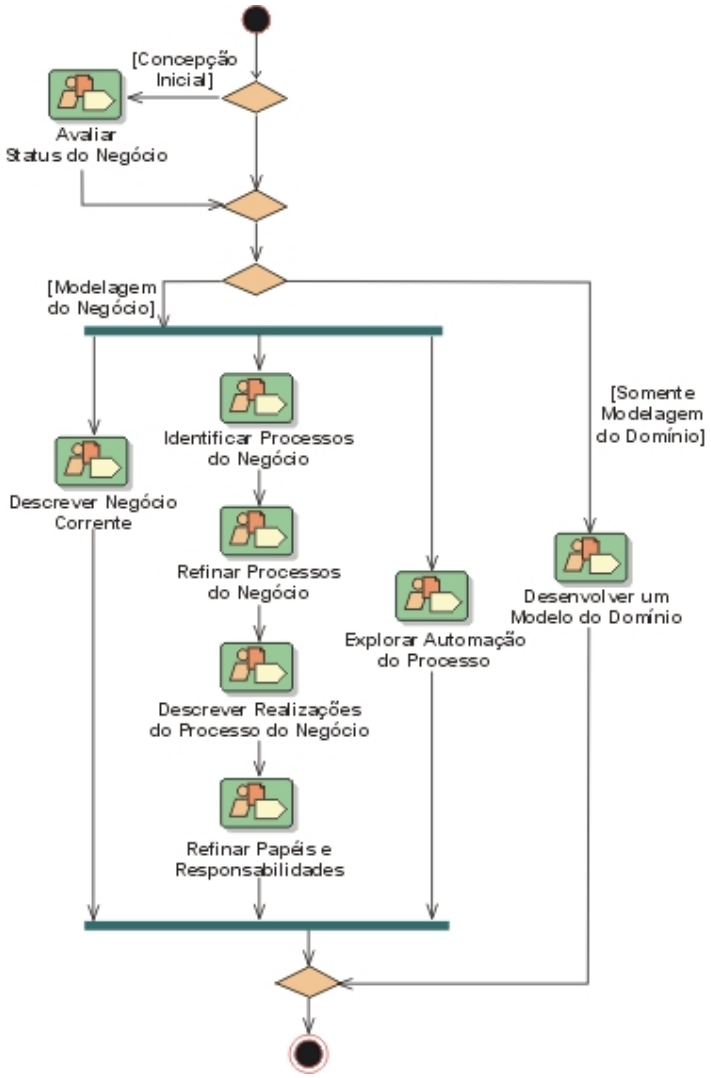


Figura 4-3 Fluxo de Modelagem de Negócios

Este fluxo descreve como desenvolver uma visão da organização, e baseado nesta visão são definidos processos, papéis, e responsabilidades desta organização em um Modelo de Casos de Uso de Negócio e um Modelo de Objetos de Negócio. Para complementar estes modelos, são produzidos os artefatos Documento de Especificação Complementar de Negócio e Glossário de Negócio.

O Modelo de Casos de Uso de Negócio representa as funções que o negócio deve desempenhar. Este modelo é usado como entrada essencial para identificação de papéis e produtos na organização e deve ser desenvolvido se houver a necessidade de clarificação do contexto do negócio do sistema a ser desenvolvido. O Modelo de Objetos de Negócio descreve a realização dos Casos de Uso de negócio. O Documento de Especificação Complementar de Negócio apresenta quaisquer definições necessárias do negócio não incluídas no Modelo de Casos de Uso de Negócio ou no Modelo de Objetos de Negócio. O Glossário de Negócio define termos importantes usados na modelagem de negócio do projeto.

Este fluxo relaciona-se com outros fluxos da seguinte forma: o fluxo de Requisitos utiliza os modelos de negócio como uma importante entrada para entendimento dos requisitos do sistema; o fluxo de Análise & Projeto utiliza entidades de negócio como uma entrada para identificação de classes de entidade no Modelo de Projeto.

4.5.2 Fluxo de Requisitos

Os propósitos deste fluxo, apresentado na Figura 4-4, são os seguintes: estabelecimento e manutenção de um acordo com os clientes e outros *stakeholders* sobre o que o sistema deve fazer; proporcionar aos desenvolvedores do sistema um melhor entendimento dos requisitos do sistema; definição dos limites do sistema; fornecer uma base para o planejamento de conteúdos técnicos das iterações; fornecer uma base para estimação de custos e tempo para o desenvolvimento do sistema; definição das interfaces gráficas do sistema, focando nas necessidades e objetivos dos usuários.

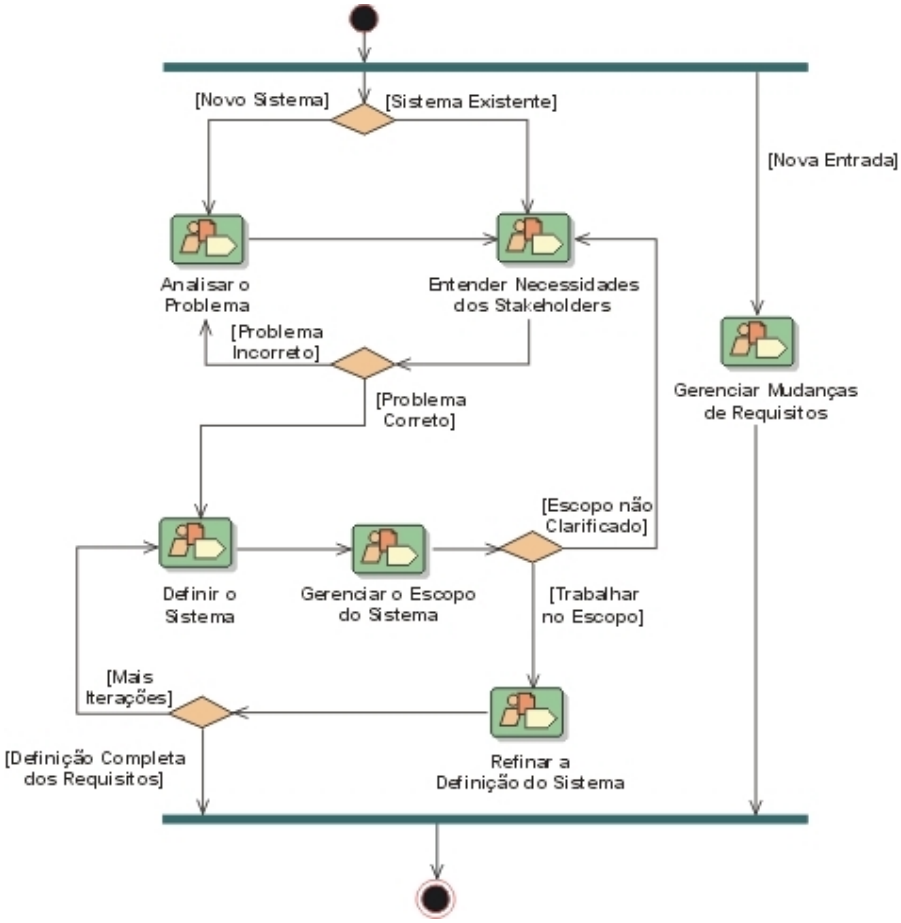


Figura 4-4 Fluxo de Requisitos

Os principais artefatos produzidos por este fluxo, são os seguintes: Documento de Visão, Documento de Necessidades dos *Stakeholders*, Modelo de Casos de Uso, Documento de Especificação Complementar, Glossário e um protótipo de interfaces gráficas.

O Documento de Visão apresenta uma visão geral dos principais requisitos do projeto e características do sistema. O Documento de Necessidades dos *Stakeholders* contém uma lista sobre o que os diferentes *stakeholders* do projeto esperam que o sistema faça. O Modelo de Casos de Uso descreve as funcionalidades do sistema e é usado como entrada essencial para as atividades de análise, projeto e teste. O Documento de Especificação Complementar descreve os requisitos não-funcionais do sistema. O Glossário define termos importantes usados no projeto. O protótipo de interfaces gráficas serve para expor e testar a funcionalidade e a usabilidade do sistema antes da etapa de projeto.

Este fluxo relaciona-se com outros fluxos da seguinte forma: o fluxo de Modelagem de Negócio fornece um contexto organizacional para o sistema; o fluxo de Análise & Projeto tem como entrada primária os requisitos; o fluxo de Teste baseia-se no Modelo de Casos de Uso e no documento de Especificação Complementar para o planejamento dos testes do sistema.

4.5.3 Fluxo de Análise & Projeto

Os propósitos deste fluxo, apresentado na Figura 4-5, são os seguintes: transformar os requisitos em um projeto do sistema; definir uma arquitetura robusta para o sistema; adaptar o projeto para o ambiente de implementação.

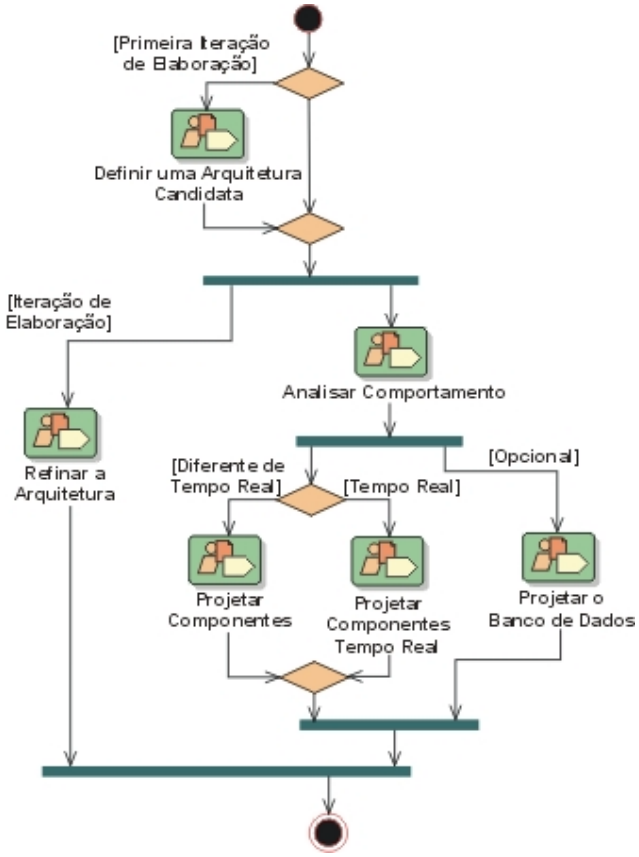


Figura 4-5 Fluxo de Análise & Projeto

Os principais artefatos produzidos por este fluxo são os seguintes: Modelo de Análise, Modelo de Projeto, Documento de Arquitetura de Software, Realizações de Caso de Uso e Modelo de Dados.

O Modelo de Análise é o modelo conceitual do sistema contendo as classes básicas e seus relacionamentos, é o resultado da atividade de análise dos Casos de Uso. O Modelo de Projeto descreve as realizações dos Casos de Uso, e serve como entrada essencial para as atividades de implementação e teste. O Documento de Arquitetura de Software fornece uma visão geral da arquitetura do sistema, através de visões arquiteturais que descrevem diferentes aspectos do sistema. As Realizações de Caso de Uso descrevem, através de diagramas de interação, como Casos de Uso são realizados dentro do Modelo de Projeto, em termos de colaboração de objetos. O Modelo de Dados descreve a representação lógica e física dos dados persistentes do sistema.

Este fluxo relaciona-se com outros fluxos da seguinte forma: o fluxo de Modelagem de Negócio fornece um contexto organizacional para o sistema; o fluxo de Requisitos fornece a entrada primária para Análise e Projeto; o fluxo de Teste testa o sistema projetado durante a Análise e Projeto.

4.5.4 Fluxo de Implementação

Os propósitos deste fluxo, apresentado na Figura 4-6, são os seguintes: definir a organização do código, em termos de subsistemas de implementação organizados em camadas; implementar classes e objetos em termos de componentes; testar os componentes desenvolvidos como unidades; integrar os resultados produzidos pelos implementadores individuais (ou times), em um sistema executável.

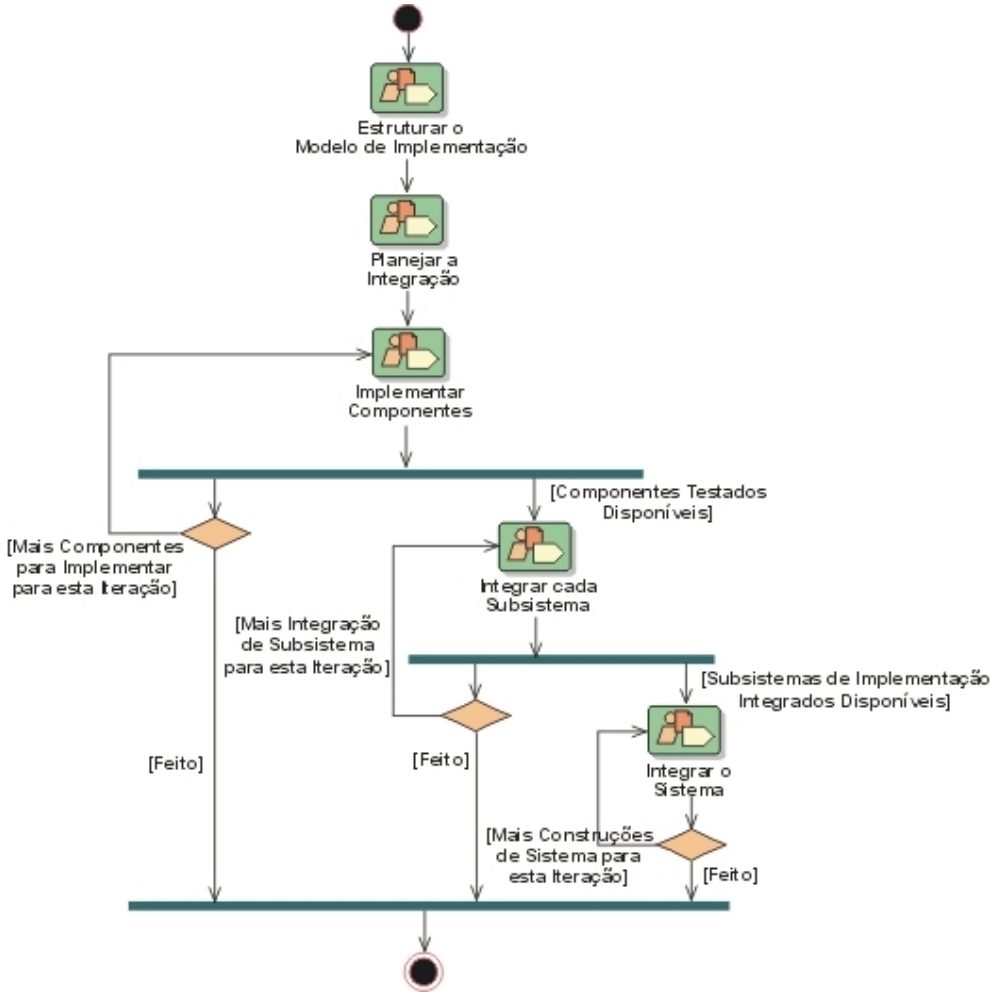


Figura 4-6 Fluxo de Implementação

Os principais artefatos produzidos por este fluxo, são os seguintes: Modelo de Implementação, Plano de Construção da Integração, Componentes e Subsistemas de Implementação.

O Modelo de Implementação é uma coleção de subsistemas de implementação e seus componentes, necessários para construir e gerenciar o sistema no ambiente de execução. O Plano de Construção da Integração fornece um plano detalhado para integração dos componentes e subsistemas em uma iteração. Um Componente representa uma peça de código de software (fonte, binário ou executável) ou um agregado de outros componentes, como uma aplicação com vários executáveis. Um Subsistema de Implementação é uma coleção de componentes e outros subsistemas de implementação, usado para estruturar o Modelo de Implementação.

4.5.5 Fluxo de Teste

Os propósitos deste fluxo, apresentado na Figura 4-7, são os seguintes: verificar a integração entre objetos; verificar a integração formal de todos os componentes do software; verificar se todos os requisitos foram corretamente implementados; identificar e garantir que defeitos sejam resolvidos antes da implementação do software.

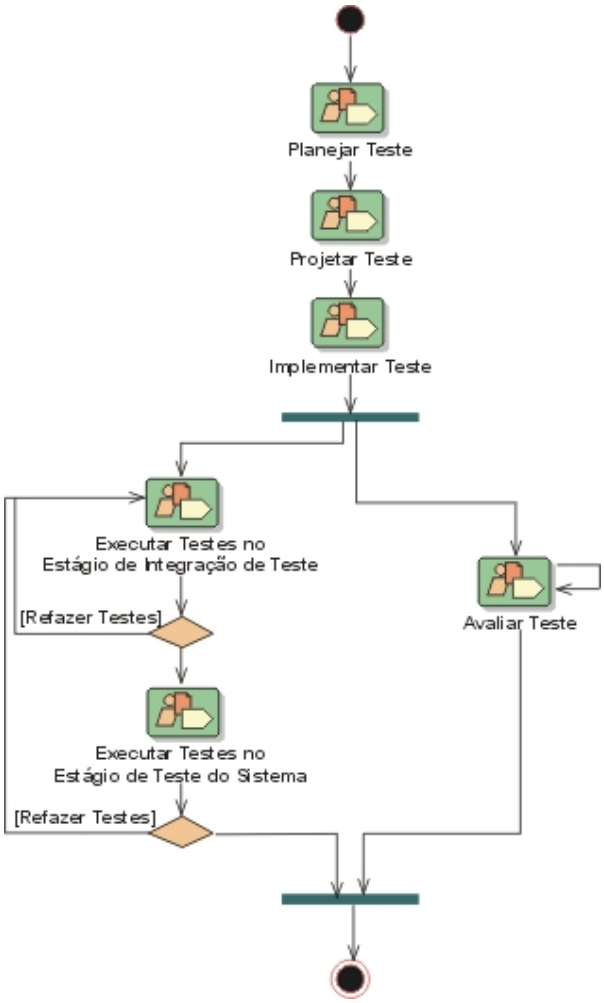


Figura 4-7 Fluxo de Teste

Os principais artefatos produzidos por este fluxo, são os seguintes: Plano de Teste, Procedimento de Teste, Caso de Teste, Script de Teste, Modelo de Teste e Sumário de Avaliação de Teste.

O Plano de Teste contém informações sobre os objetivos dos testes e identifica estratégias e recursos a serem usados para implementar e executar os testes. Um Caso de Teste é um conjunto de entradas para o teste, condições de execução, e resultados esperados.

Um Procedimento de Teste é um conjunto de instruções detalhadas para configuração, execução e avaliação de resultados de um Caso de Teste. Os Scripts de Teste são execuções automatizadas de um Procedimento de Teste. O Modelo de Teste é uma representação do que será testado e como será testado. O Sumário de Avaliação de Teste organiza e apresenta os resultados dos testes.

4.5.6 Fluxo de Implantação

O propósito deste fluxo, apresentado na Figura 4-8, é a produção de versões do sistema e a entrega do software para os usuários finais. É representado pela execução de várias atividades, tais como, produzir versões do software, empacotar, distribuir e instalar o software, fornecer ajuda e assistência aos usuários, planejamento e execução de beta testes, migração de software ou dados já existentes e aceitação formal.

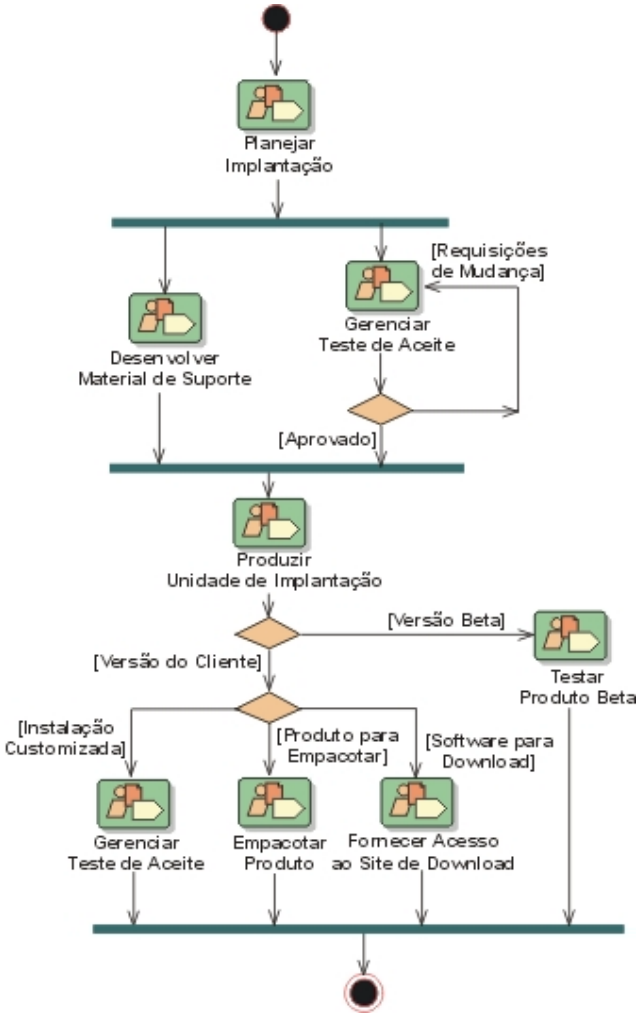


Figura 4-8 Fluxo de Implantação

Os principais artefatos produzidos por este fluxo são: Plano de Implantação, Material de Suporte ao Usuário Final e Materiais de Treinamento.

O Plano de Implantação determina como será a transição do produto para os usuários, com o propósito de garantir que os usuários recebam corretamente o sistema. Neste plano são consideradas questões sobre a instalação, distribuição, treinamento e suporte. O Material de Suporte ao Usuário Final assiste os usuários finais no aprendizado, uso, operação e manutenção do produto. Os Materiais de Treinamento são os materiais que são usados nos programas de treinamento ou cursos para assistir os usuários finais com o uso, operação e/ou manutenção do produto.

4.5.7 Fluxo de Gerenciamento de Configuração e Mudanças

Este fluxo controla as mudanças e mantém a integridade dos artefatos do projeto. Neste fluxo são identificados os itens de configuração, restrições a mudanças nestes itens, verificação de mudanças feitas nestes itens e a definição e gerenciamento das configurações dos itens, durante o processo de desenvolvimento. Os métodos, processos e ferramentas usados para fornecer gerenciamento de configuração e mudança para uma organização podem ser considerados como o sistema de CM (*Configuration Management*) da organização.

Um sistema de CM manuseia informações importantes sobre os processos de desenvolvimento, implantação e manutenção, além de conservar o acervo base de artefatos potencialmente reusáveis, resultantes da execução destes processos. Este sistema é essencial para o controle dos numerosos artefatos produzidos pelas várias pessoas que trabalham em um mesmo projeto. O controle ajuda a evitar que os artefatos resultantes não estejam conflitantes em relação a questões como atualização simultânea, notificação limitada[4] e múltiplas versões.

4.5.8 Fluxo de Gerenciamento de Projeto

Os propósitos deste fluxo são os seguintes: fornecer um *framework* para gerenciamento de projetos de software; fornecer guias práticos para planejamento, recrutamento de pessoal, execução e monitoração de projetos; fornecer um *framework* para o gerenciamento de riscos.

[4] Quando um problema é corrigido em artefatos compartilhados por vários desenvolvedores, e alguns deles não são notificados da mudança.

Este fluxo foca principalmente nos aspectos importantes de um processo de desenvolvimento iterativo: gerenciamento de riscos, planejamento de um projeto iterativo e monitoração do progresso de um projeto iterativo (métricas). Entretanto, este fluxo não atende a todos os aspectos de gerenciamento de projeto, tais como, gerenciamento de pessoal, gerenciamento de orçamento e gerenciamento de contratos.

4.5.9 Fluxo de Ambiente

Este fluxo foca nas atividades necessárias para configurar o processo para um projeto. O propósito das atividades de ambiente é fornecer a organização e um ambiente (processos e ferramentas) de desenvolvimento do software, que darão suporte a equipe de desenvolvimento. Este fluxo não atende questões como seleção, aquisição e construção de ferramentas; e manutenção do ambiente de desenvolvimento.

4.6 Considerações Finais

Neste capítulo foram apresentadas as características do *Rational Unified Process*, bem como o detalhamento de suas fases e fluxos. O RUP é um processo de software genérico para atender o desenvolvimento de vários tipos de aplicação. Entretanto, esta generecidade faz com que características específicas de alguns tipos de aplicação, como aplicações Web, não sejam devidamente contempladas. Diante deste panorama, o próximo capítulo apresenta uma extensão do fluxo de Análise e Projeto para atender mais apropriadamente o desenvolvimento de aplicações Web.

Capítulo 5

Extensão do Fluxo de Análise e Projeto do RUP para o Desenvolvimento de Aplicações Web

O RUP é uma metodologia genérica que pode ser especializada para várias classes de sistemas de software e áreas de aplicação. O desenvolvimento de aplicações Web apresenta características substancialmente diferentes do desenvolvimento de aplicações tradicionais, como questões sobre elicitação de requisitos, interface do usuário, arquitetura da aplicação, diversidade tecnológica, estratégias de teste e implantação; conforme apresentado por A.Araújo [Araújo 01].

A maioria das diferenças entre o desenvolvimento de aplicações Web e aplicações tradicionais se concentra na etapa de Análise e Projeto. Nesta etapa, R.Pressman define em [Pressman 92] que, "... tomam-se decisões que em última análise afetarão o sucesso da implementação do software e, o que é igualmente importante, a facilidade com que o software será mantido, tornando-o, assim, um passo fundamental do desenvolvimento de software. O projeto é o lugar onde a qualidade é fomentada, servindo como molde para todos os passos de desenvolvimento que se seguirão...".

Diante da necessidade de um método especifico para o desenvolvimento de aplicações Web, da importância da etapa de Análise e Projeto para o sucesso do desenvolvimento de uma aplicação e da característica do RUP de poder ser adaptado para os diferentes tipos de aplicação, este capítulo apresenta uma extensão do fluxo de Análise e Projeto do RUP para o desenvolvimento de aplicações Web.

5.1 Visão Geral

O propósito do fluxo de Análise e Projeto é transformar os requisitos em um Modelo de Projeto implementável, evoluir uma arquitetura robusta para o sistema e adaptar o projeto de acordo com o ambiente de implementação. Este fluxo foi estendido para satisfazer as necessidades especificas de aplicações Web, principalmente no que se refere aos aspectos de navegação e de apresentação da aplicação.

A extensão do fluxo de Análise e Projeto do RUP para desenvolvimento de aplicações Web baseia-se no framework de Análise e Projeto proposto por A.Araújo [Araújo 01] e em extensões de UML propostas por J.Connalen [Conallen 99b] e por N.Koch [Koch 01].

5.2 Fluxo de Análise e Projeto do RUP Estendido

O fluxo de Análise e Projeto do RUP foi estendido para o domínio de aplicações Web. A extensão do fluxo atende a aspectos específicos, não observados originalmente, para criação da Camada de Apresentação de uma aplicação Web desenvolvida no estilo arquitetural de camadas. Além disso, foram feitas recomendações para atividades já existentes deste fluxo, para satisfazer mais apropriadamente o desenvolvimento de aplicações Web.

O fluxo de Análise e Projeto do RUP foi estendido e adaptado para considerar mais apropriadamente o desenvolvimento de aplicações Web sem perder, contudo, as características de adequabilidade a outros tipos de aplicação e a generecidade do processo.

Na Figura 5-1, um diagrama de atividades da UML é utilizado para representar este fluxo, sendo que cada atividade do diagrama corresponde a um subfluxo, que é uma abstração da execução de atividades do fluxo de Análise e Projeto.

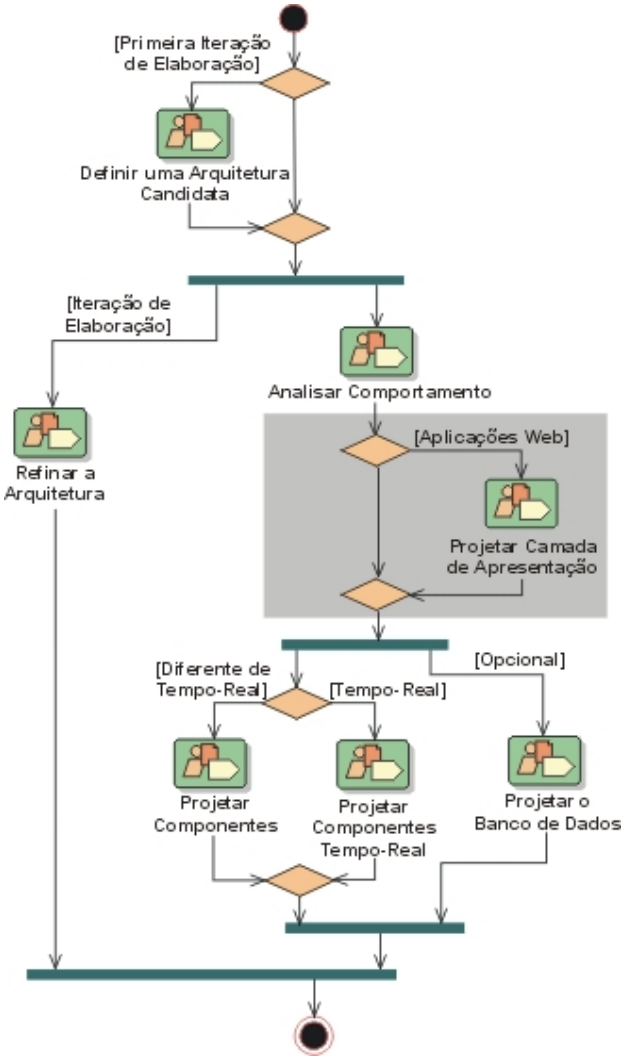


Figura 5-1 Fluxo de Análise e Projeto do RUP Estendido para Aplicações Web

A primeira iteração de Elaboração foca na criação de uma arquitetura inicial para o sistema, através do subfluxo “Definir uma Arquitetura Candidata”, para fornecer o ponto de partida do principal trabalho de análise. Se a arquitetura já foi produzida nas iterações anteriores ou em projetos anteriores, o foco do trabalho muda para o refinamento da arquitetura, através do subfluxo “Refinar a Arquitetura”, e análise do comportamento, através da criação de um conjunto inicial de elementos que fornecem comportamento apropriado, representado pelo subfluxo “Analisar Comportamento”. Se a aplicação for baseada na Web, o novo subfluxo proposto “Projetar Camada de Apresentação” é executado.

Após os elementos iniciais serem identificados, eles são gradativamente refinados. Os subfluxos “Projetar Componentes” e “Projetar Componentes de Tempo Real” produzem um conjunto de componentes que fornecem comportamento apropriado para satisfazer os

requisitos do sistema. Em paralelo, as persistências são tratadas no subfluxo “Projetar Banco de Dados”.

5.3 Subfluxos do Fluxo de Análise e Projeto Estendido

Os subfluxos do fluxo de Análise e Projeto do RUP estendido (Figura 5-1) para o desenvolvimento de aplicações Web apresentam a seqüência de execução das atividades e seus objetivos, com o propósito de garantir subsídios para o fluxo de Implementação. Nas subseções a seguir, são apresentados resumos dos subfluxos originais, bem como uma síntese do novo subfluxo proposto para o projeto da camada de apresentação de aplicações Web. As atividades executadas (mencionadas) nesses subfluxos são apresentadas na seção 5.4.

5.3.1 Definir uma Arquitetura Candidata

A Figura 5-2 apresenta o detalhamento deste subfluxo, através da identificação das atividades, responsáveis, artefatos de entrada e artefatos produzidos.

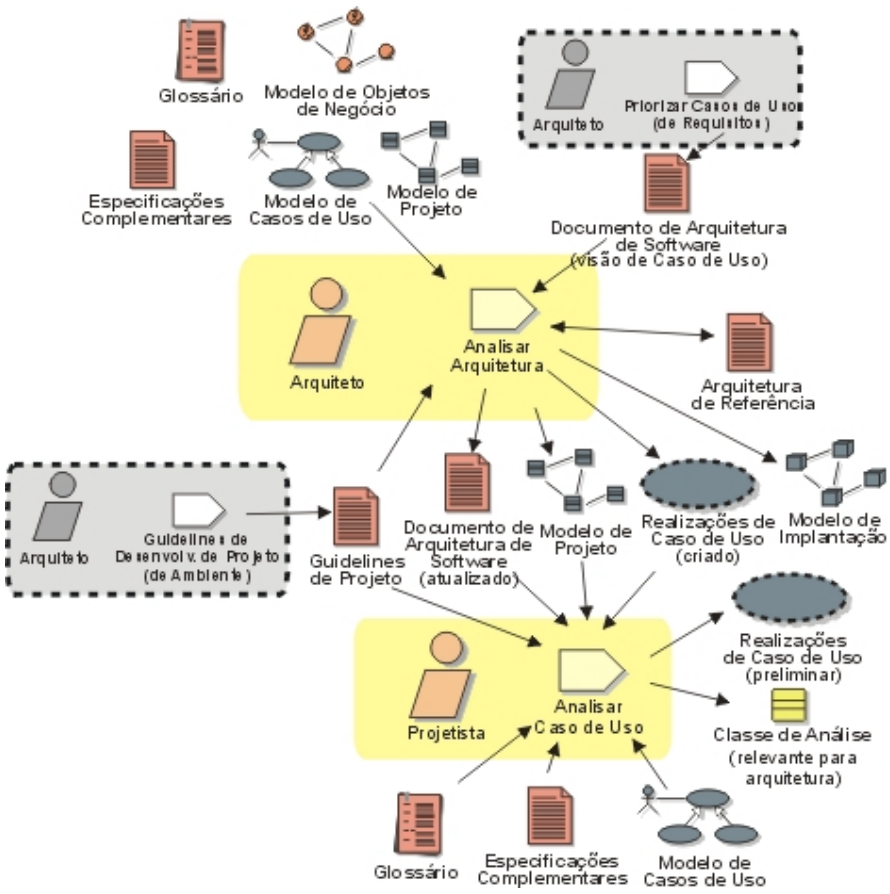


Figura 5-2 Subfluxo Definir uma Arquitetura Candidata

Neste subfluxo é criado um esqueleto inicial da arquitetura do sistema, é definido um conjunto inicial de elementos relevantes para a arquitetura, para serem usados como base para a análise, é definido um conjunto inicial de Mecanismos de Análise (padrão que constitui uma solução comum para um problema comum, independente da implementação, como por exemplo: mecanismos para manipular persistência e erros ou falhas, e comunicação entre processos.), é definida a organização do sistema, e são definidas as Realizações de Casos de Uso (descreve como um Caso de Uso é realizado dentro do Modelo de Projeto, em termos de colaboração de objetos, expressado por diagramas de interação de UML) para serem utilizadas na iteração corrente. Além disso, são identificadas as Classes de Análise (representa um Modelo Conceitual para "coisas no sistema que tenham responsabilidades e comportamento") dos Casos de Uso relevantes para a arquitetura do sistema, e são atualizadas as Realizações de Casos de Uso com as interações das Classes de Análise.

O subfluxo inicia com a definição de uma arquitetura inicial para o sistema, através da atividade "Analisar Arquitetura", depois os Casos de Uso relevantes para a arquitetura são identificados e para cada um deles a atividade "Analisar Caso de Uso" é realizada. O subfluxo termina com a atualização da arquitetura para refletir as adaptações requeridas para satisfazer o novo comportamento do sistema e para corrigir problemas arquiteturais identificados.

5.3.2 Refinar a Arquitetura

A Figura 5-3 apresenta o detalhamento deste subfluxo, através da identificação das atividades, responsáveis, artefatos de entrada e artefatos produzidos.

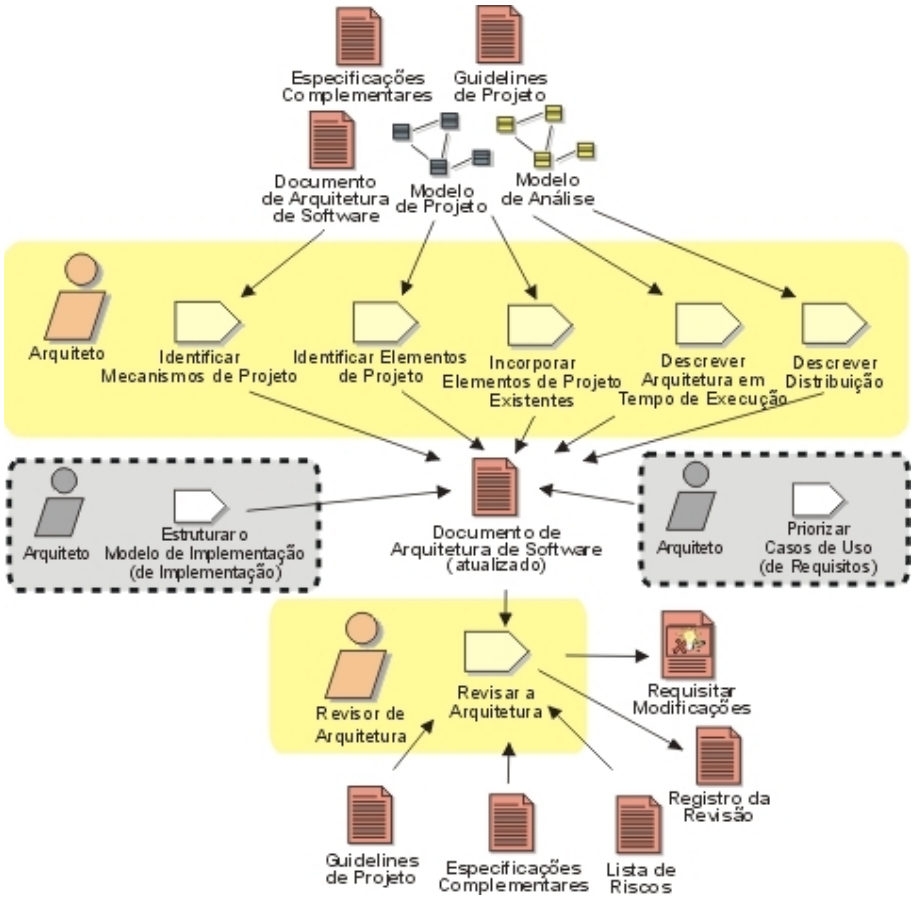


Figura 5-3 Subfluxo Refinar a Arquitetura

Neste subfluxo é fornecida a transição natural das atividades de análise para as atividades de projeto, através da identificação dos elementos de projeto (refinamento dos elementos de análise) apropriados e da identificação dos Mecanismos de Projeto (refinamento dos Mecanismos de Análise) apropriados. A arquitetura é mantida consistente e íntegra, possibilitando: a integração dos novos elementos de projeto identificados na iteração corrente com os elementos de projeto pré-existentes, e o máximo reuso de componentes disponíveis. A organização da execução do sistema é descrita e o Modelo de Implementação[5] é organizado para fazer a transição entre projeto e implementação.

[5] Coleção de subsistemas de implementação e seus componentes.

O subfluxo inicia com foco nas atividades “Identificar Mecanismos de Projeto” e “Identificar Elementos de Projeto”, e com grande ênfase na atividade “Incorporar Elementos de Projeto Existentes” para garantir que novos elementos não dupliquem as funcionalidades de elementos existentes. Após isso, conceitos de concorrência e distribuição são introduzidos nas atividades “Descrever a Arquitetura em Tempo de Execução” e “Descrever Distribuição”, respectivamente. O subfluxo termina com a execução da atividade “Revisar a Arquitetura”.

5.3.3 Analisar Comportamento

A Figura 5-4 apresenta o detalhamento deste subfluxo, através da identificação das atividades, responsáveis, artefatos de entrada e artefatos produzidos.

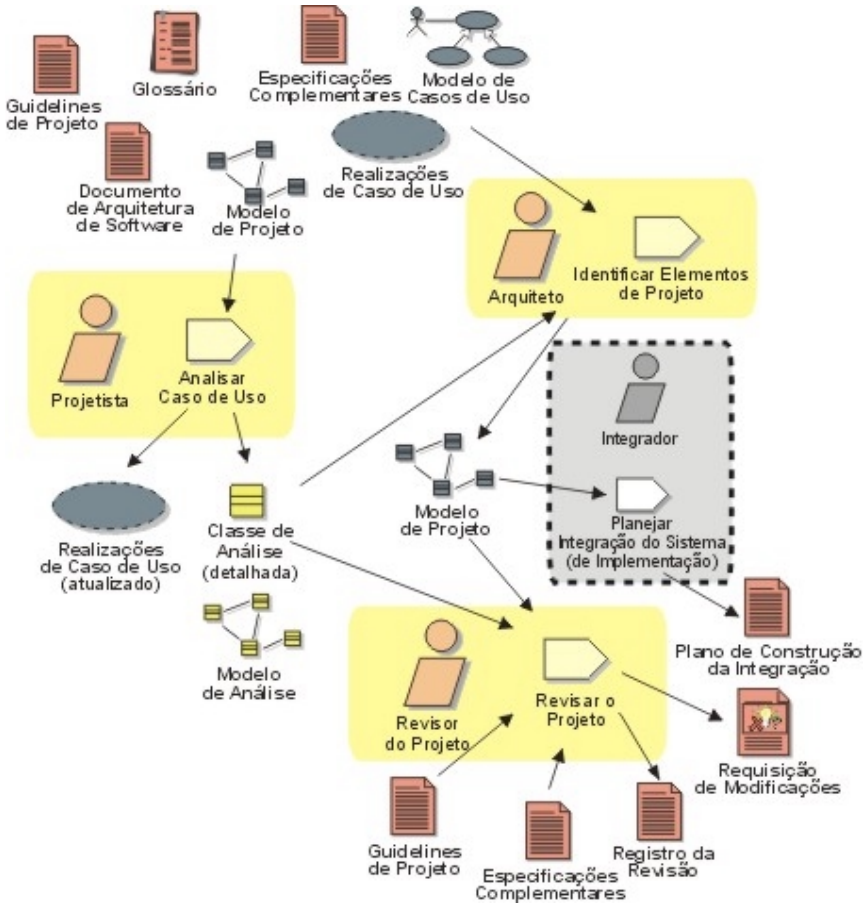


Figura 5-4 Subfluxo Analisar Comportamento

Neste subfluxo, as descrições comportamentais do sistema fornecidas pelos Casos de Uso são transformadas em um conjunto de elementos sobre os quais o projeto deve se basear.

O subfluxo inicia com *workshops* para análise dos Casos de Uso, realizados na atividade "Analisar Caso de Uso". Posteriormente, os projetistas responsáveis pela análise dos Casos de Uso em conjunto com os arquitetos do sistema fundem os resultados dos *workshops* na atividade "Identificar Elementos de Projeto". O subfluxo termina com a execução da atividade "Revisar o Projeto".

5.3.4 Projetar Camada de Apresentação

A Figura 5-5 apresenta o detalhamento deste subfluxo, através da identificação das atividades, responsáveis, artefatos de entrada e artefatos produzidos.

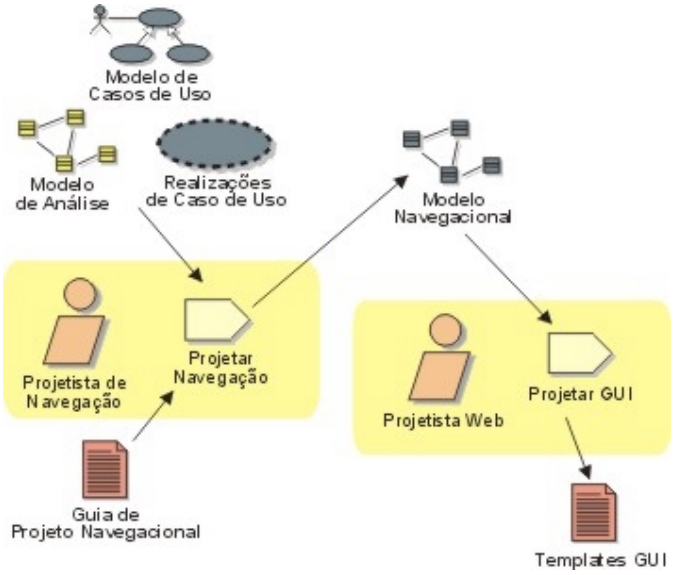


Figura 5-5 Subfluxo Projetar Camada de Apresentação

Este subfluxo foi criado com o propósito de atender aos aspectos de navegação e de apresentação, relacionados à criação da Camada de Apresentação de uma aplicação Web, não observados no fluxo de Análise e Projeto original. Os objetivos deste subfluxo são identificar a navegação das funcionalidades da aplicação e criar subsídios para o projeto das interfaces gráficas do usuário.

A execução do subfluxo inicia com o processo de criação do Modelo Navegacional da aplicação, através da realização da atividade "Projetar Navegação" cujo objetivo é identificar como o usuário caminha (navega) na aplicação para utilizar as funcionalidades do sistema. O subfluxo termina com a realização da atividade "Projetar GUI", com o objetivo de projetar as interfaces gráficas do usuário com base no Modelo Navegacional.

5.3.5 Projetar Componentes

A Figura 5-6 apresenta o detalhamento deste subfluxo, através da identificação das atividades, responsáveis, artefatos de entrada e artefatos produzidos.

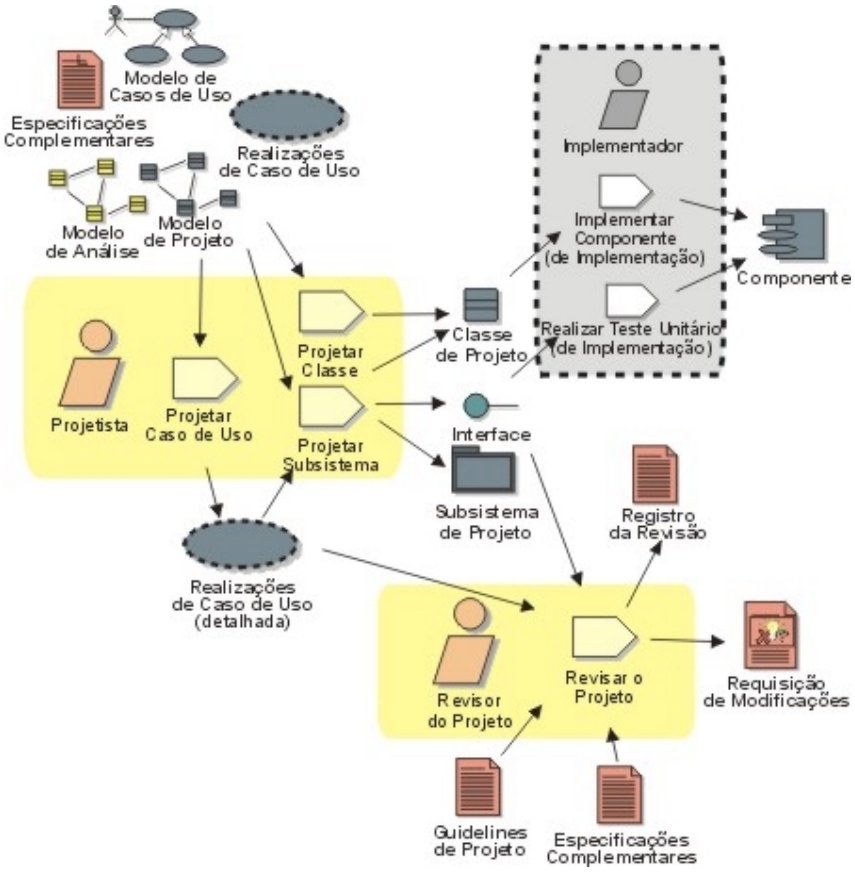


Figura 5-6 Subfluxo Projetar Componentes

Neste subfluxo, as definições dos elementos de projeto são refinadas. As Realizações de Casos de Uso são refinadas e atualizadas com base nos novos elementos de projeto identificados. O projeto é revisado de acordo com sua evolução. Os elementos de projeto são implementados como componentes e testados para verificar a funcionalidade e a satisfação dos requisitos no nível componente/unitário.

O subfluxo inicia com o refinamento dos elementos de projeto, através das atividades "Projetar Classe" e "Projetar Subsistema". Posteriormente, as Realizações de Casos de Uso são atualizadas para contemplar o refinamento ou criação de elementos de projeto, através da atividade "Projetar Caso de Uso". O subfluxo termina com a execução da atividade "Revisar o

Projeto", para garantir que todos os comportamentos requeridos para o sistema estejam suportados e que os elementos de projeto estejam prontos para implementação.

5.3.6 Projetar Componentes de Tempo-Real

A Figura 5-7 apresenta o detalhamento deste subfluxo, através da identificação das atividades, responsáveis, artefatos de entrada e artefatos produzidos.

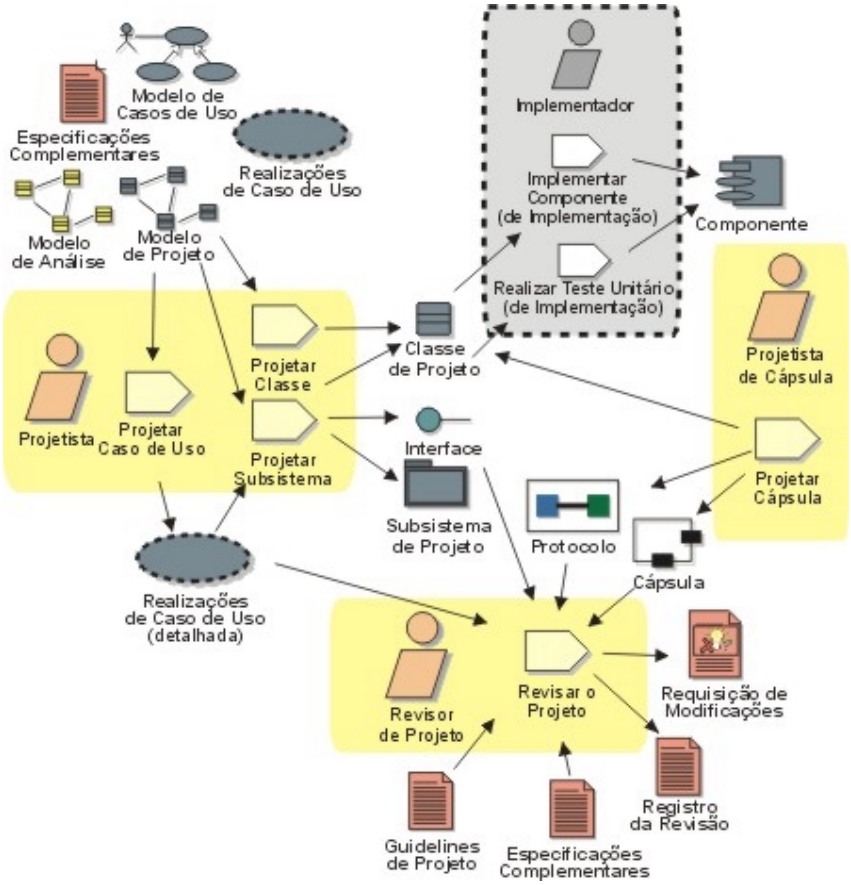


Figura 5-7 Subfluxo Projetar Componentes de Tempo Real

O subfluxo é similar a "Projetar Componentes", apresentado na subseção 5.3.5. A diferença é que este subfluxo é aplicado a projetos que utilizam o artefato Cápsula[6] como elemento de projeto primário, dentro do contexto de sistemas de tempo real ou sistemas reativos.

[6] Padrão de projeto que representa um *thread* de controle encapsulado no sistema, representa a unidade primária de concorrência no sistema.

5.3.7 Projetar Banco de Dados

A Figura 5-8 apresenta o detalhamento deste subfluxo, através da identificação das atividades, responsáveis, artefatos de entrada e artefatos produzidos.

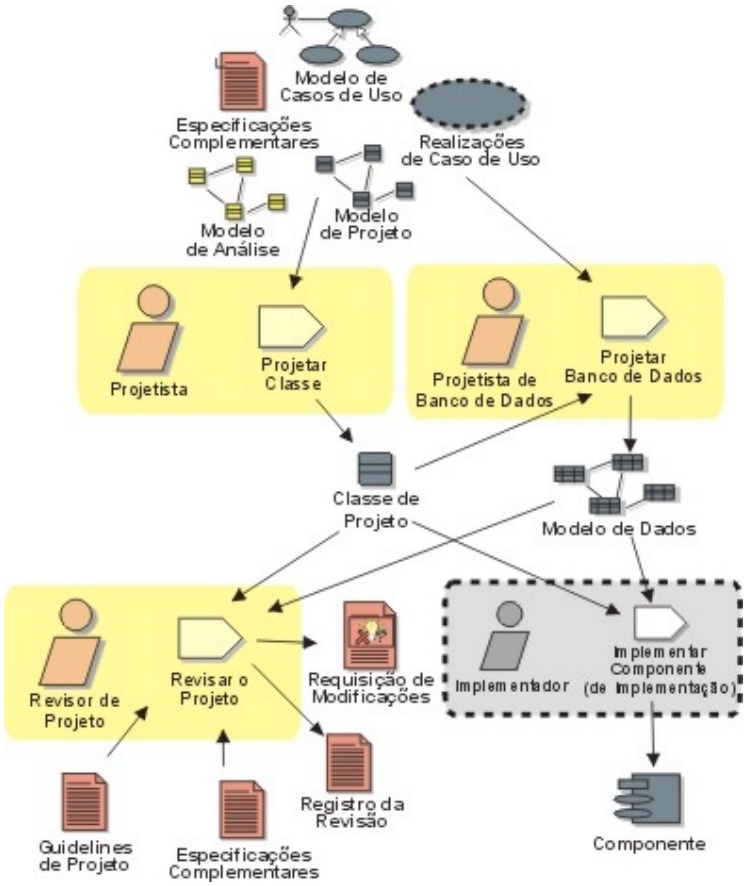


Figura 5-8 Subfluxo Projetar Banco de Dados

Neste subfluxo são identificadas as classes persistentes do projeto, são projetadas as estruturas de banco de dados apropriadas para armazenar estas classes, e são definidos mecanismos e estratégias para armazenar e recuperar dados persistentes de modo a satisfazer o critério de performance do sistema.

O subfluxo inicia com a identificação das Classes de Projeto persistentes, através da atividade “Projetar Classe”. Posteriormente, um Modelo de Dados é criado, através da atividade “Projetar Banco de Dados”, e o subfluxo termina com a execução da atividade “Revisar o Projeto”.

5.4 Atividades do Fluxo de Análise e Projeto Estendido

As atividades originais do fluxo de Análise e Projeto do RUP foram estendidas com recomendações para satisfazer mais apropriadamente o desenvolvimento de aplicações Web, seguindo o framework de Análise e Projeto proposto por A.Araújo [Araújo 01] apresentado no Anexo C. Além disso, foram criadas as atividades "Projetar Navegação" e "Projetar GUI", com o propósito de estruturar a criação da Camada de Apresentação de aplicações Web.

Nas subseções a seguir, as atividades do fluxo de Análise e Projeto estendido são apresentadas. O responsável pela atividade é identificado e os propósitos da atividade, juntamente com as extensões (recomendações) para o desenvolvimento de aplicações Web, se houver, são descritos.

5.4.1 Analisar Arquitetura

Nesta atividade o Arquiteto deve definir uma arquitetura candidata para o sistema, baseado na experiência obtida em sistemas similares; definir padrões arquiteturais, mecanismos chave e convenções de modelagem para o sistema (tipos de diagramas e elementos de modelagem que devem ser usados, e as regras para o uso); definir a estratégia de reuso e fornecer entrada para o processo de planejamento.

O framework de Análise e Projeto proposto por A.Araújo [Araújo 01], conforme apresentado no Anexo C – Atividade Analisar Arquitetura, recomenda entre outras coisas a utilização do artefato Guidelines de Projeto Web para auxiliar na execução desta atividade para aplicações Web. Este artefato fornece algumas recomendações sobre pontos importantes para a definição inicial da arquitetura, como definição de camadas, padrões de distribuição, Mecanismos de Análise e identificação de abstrações chave. Além disso, este artefato pode ser atualizado durante todo o projeto e através do portfólio de projetos da organização, registrando novas recomendações ou boas práticas, que forem identificadas para o desenvolvimento de aplicações Web.

5.4.2 Identificar Mecanismos de Projeto

Nesta atividade o Arquiteto deve refinar os Mecanismos de Análise em Mecanismos de Projeto, baseado nas restrições impostas pelo ambiente de implementação.

As aplicações Web têm sido desenvolvidas usando uma combinação de tecnologias de orientação a objetos, cliente/servidor e Internet. Surge, então, um grande número de opções de infra-estrutura que levam a um ilimitado número de opções tecnológicas. Desta forma, a implementação de uma aplicação Web envolve um grande número de tecnologias e restrições, levando a um ambiente de implementação dos mais diversos.

Para aplicações Web, conforme apresentado no Anexo C – Atividade Projetar Arquitetura – Passo Identificar Mecanismos de Projeto, esta atividade deve dar maior importância às restrições antes do refinamento dos Mecanismos de Análise em Mecanismos de Projeto, pois, assim sendo, o risco de opções tecnológicas conflitantes ou pouco adequadas a serem utilizadas para implementação da aplicação é reduzido.

5.4.3 Identificar Elementos de Projeto

Nesta atividade o Arquiteto deve analisar as interações das Classes de Análise para identificar elementos do Modelo de Projeto.

Esta atividade já atende às necessidades de aplicações Web, pois como as aplicações tradicionais, o desenvolvimento de aplicações Web é baseado nos Casos de Uso que descrevem as funcionalidades da aplicação e originam as Classes de Análise.

5.4.4 Incorporar Elementos de Projeto Existentes

Nesta atividade o Arquiteto deve analisar interações entre as Classes de Análise para encontrar Interfaces, Classes de Projeto e Subsistemas; refinar a arquitetura, incorporando reuso onde possível; identificar soluções comuns para problemas usuais de projeto; e incluir elementos do Modelo de Projeto significantes para a arquitetura na seção Visão Lógica do Documento de Arquitetura de Software[7].

Para aplicações Web, conforme apresentado no Anexo C – Atividade Projetar Arquitetura – Passo Identificar Oportunidades de Reuso, esta atividade deve dar maior importância ao reuso e ao desenvolvimento baseado em componentes para aplicações Web, devido ao tempo de desenvolvimento de uma aplicação Web (*Web Time*), que geralmente é crítico.

5.4.5 Descrever a Arquitetura em Tempo de Execução

Nesta atividade o Arquiteto deve analisar requisitos de concorrência para identificar processos, identificar mecanismos de comunicação entre processos, alocar recursos de coordenação entre processos, identificar ciclos de vida do processo, e distribuir elementos do modelo entre os processos.

Este atividade, conforme apresentado no Anexo C – Atividade Descrever Concorrência, já atende às necessidades de aplicações Web, pois a criação de processos, a alocação de funcionalidades a estes processos e o seu gerenciamento é uma tarefa comum a aplicações que utilizam a noção de concorrência, sejam elas baseadas na Web ou não.

5.4.6 Descrever a Distribuição

Nesta atividade o Arquiteto deve descrever como a funcionalidade do sistema é distribuída através de nós físicos, sendo requerida apenas para sistemas distribuídos. A distribuição de processos através de dois ou mais nós requer um exame dos padrões de comunicação entre processos do sistema. O alcance de benefícios reais para a distribuição requer trabalho e planejamento cuidadoso. Uma aplicação Web é naturalmente uma aplicação distribuída, que tende a saturar a rede rapidamente caso não seja bem projetada.

A descrição da distribuição em aplicações Web é uma atividade crítica. A Internet é uma rede bastante complexa e qualquer sistema que a utilize deve considerar uma série de questões para que se tenha performance, segurança, confiabilidade e disponibilidade necessárias.

Para aplicações Web, conforme apresentado no Anexo C – Atividade Descrever Distribuição, esta atividade deve considerar questões como: desempenho do *backbone*, balanceamento de carga, segurança em nível de rede (*firewall* e *proxy*), espelhamento e replicação, redundância de dados e de processamento, interoperabilidade e interface com outros sistemas.

[7] Fornece uma visão geral compreensiva da arquitetura do sistema, usando diferentes visões arquiteturais para descrever diferentes aspectos do sistema.

5.4.7 Revisar a Arquitetura

Nesta atividade o Revisor da Arquitetura deve fazer o levantamento de alguns riscos não percebidos ou não conhecidos; detectar alguma falha no projeto arquitetural (defeitos arquiteturais são mais difíceis de detectar e mais prejudiciais em longo prazo); detectar algum conflito entre os requisitos e a arquitetura; avaliar uma ou mais qualidades arquiteturais específicas como desempenho, confiabilidade, modificabilidade, segurança de acesso e segurança de dados, e identificar a oportunidades de reuso.

Para aplicações Web, conforme apresentado no Anexo C – Atividade Revisar Arquitetura, esta atividade deve realizar *checklists* para verificar os aspectos específicos que foram considerados durante o projeto da arquitetura, de modo a fomentar a qualidade e a aplicabilidade da arquitetura.

5.4.8 Analisar Caso de Uso

Nesta atividade, o Projetista deve identificar as classes que realizam um fluxo de eventos dos Casos de Uso; distribuir o comportamento dos Casos de Uso para estas classes, usando Realizações de Caso de Uso; identificar responsabilidades, atributos e associações das classes e anotar o uso de Mecanismos Arquiteturais[8].

Esta atividade é adequada para o desenvolvimento de aplicações Web, pois da mesma forma que em aplicações tradicionais, as classes que compõem a aplicação Web são determinadas a partir dos Casos de Uso e detalhadas superficialmente através da definição de atributos, responsabilidades e associações.

5.4.9 Projetar Caso de Uso

Nesta atividade, o Projetista deve refinar as Realizações de Caso de Uso em termos de interações; refinar os requisitos nas operações de Classes de Projeto; refinar os requisitos nas operações dos subsistemas e/ou suas interfaces e refinar os requisitos nas operações das cápsulas. Esta atividade apresenta-se adequada para o desenvolvimento de aplicações Web.

[8] Padrões de estruturas, padrões de comportamento, ou ambos. Representam soluções concretas para problemas encontrados freqüentemente. No RUP, é usado como um termo genérico para os mecanismos de análise, projeto e implementação.

5.4.10 Projetar Subsistema

Nesta atividade, o Projetista deve especificar os comportamentos das interfaces dos subsistemas em termos de colaborações das classes contidas no subsistema; documentar a estrutura interna do subsistema; definir realizações entre as interfaces e as classes contidas e determinar as dependências com outros subsistemas. Esta atividade apresenta-se adequada para o desenvolvimento de aplicações Web.

5.4.11 Projetar Classe

Nesta atividade, o Projetista deve garantir que as classes providenciam o comportamento que as Realizações dos Casos de Uso requerem; garantir que este comportamento é suficiente para a implementação das classes; lidar com os requisitos não-funcionais relacionados às classes e incorporar os Mecanismos de Projeto usados pelas classes. Esta atividade apresenta-se adequada para o desenvolvimento de aplicações Web.

5.4.12 Projetar Navegação

Nesta atividade, o Projetista deve criar o Modelo Navegacional da aplicação Web, com o propósito de identificar os caminhos navegacionais da aplicação, ou seja, como o usuário caminha (navega) na aplicação para utilizar as funcionalidades do sistema.

Esta atividade foi criada com o objetivo de atender aos aspectos de navegação da aplicação e auxiliar o projeto das interfaces gráficas do sistema. Esta atividade será detalhada no próximo capítulo, juntamente com o método proposto para criação do Modelo Navegacional.

5.4.13 Projetar GUI (Graphic User Interface)

Nesta atividade, o Projetista Web deve definir a aparência das interfaces gráficas com o usuário, com base no Modelo Navegacional da aplicação Web.

Esta atividade foi criada com o objetivo de melhor atender a criação das interfaces gráficas do usuário para aplicações Web, e será detalhada no próximo capítulo.

5.4.14 Projetar Banco de Dados

Nesta atividade, o Projetista de Banco de Dados deve garantir que os dados persistentes sejam armazenados com consistência e eficiência, e definir o comportamento a ser implementado no banco de dados. Esta atividade apresenta-se adequada para o desenvolvimento de aplicações Web.

5.4.15 Projetar Cápsula

Nesta atividade, o Projetista de Cápsula elabora e refina as descrições de uma cápsula (unidade primária de concorrência do sistema). Esta atividade é executada apenas para o desenvolvimento de aplicações de tempo real.

5.4.16 Revisar o Projeto

Nesta atividade, o Revisor do Projeto deve verificar se o Modelo de Projeto cumpre os requisitos do sistema e serve como uma boa base para a implementação; garantir que o Modelo de Projeto é consistente e que respeita as diretrizes gerais do projeto; e garantir que as diretrizes do projeto cumprem seus objetivos.

Ao final da execução do fluxo de Análise e Projeto, os modelos construídos ou atualizados durante a sua execução devem ser revisados. Esta revisão deve considerar os novos aspectos inseridos no fluxo para torná-lo mais apropriado ao desenvolvimento de aplicações Web.

5.5 Considerações Finais

Neste capítulo, foi apresentada a extensão do fluxo de Análise e Projeto do RUP, para atender mais apropriadamente o desenvolvimento de aplicações Web. A principal característica desta extensão foi a criação do subfluxo “Projetar Camada de Apresentação” com o objetivo de atender aos aspectos de navegação e de apresentação, bastante relevantes em se tratando de aplicações Web. Além disso, foram feitas recomendações para atividades já existentes com base no framework de Análise e Projeto proposto por A.Araújo [Araújo 01].

O próximo capítulo apresenta um detalhamento do subfluxo “Projetar Camada de Apresentação” e de suas atividades, bem como o método proposto para criação do Modelo Navegacional de uma aplicação Web.

Capítulo 6

Detalhamento do Subfluxo Projetar Camada de Apresentação

O fluxo de Análise e Projeto do RUP foi adaptado para atender mais apropriadamente o desenvolvimento de aplicações Web, conforme apresentado no capítulo 5. Esta adaptação foi baseada no framework de Análise e Projeto proposto por A.Araújo [Araújo 01], e possui como principal característica a criação de um novo subfluxo que atende aos aspectos de navegação e de apresentação de uma aplicação Web.

O novo subfluxo chamado "Projetar Camada de Apresentação" possui como principais objetivos: identificar a navegação das funcionalidades da aplicação e criar subsídios para o projeto das interfaces gráficas do usuário. A execução deste subfluxo consiste na realização das atividades Projetar Navegação, cujo objetivo é a criação do Modelo Navegacional da aplicação, e Projetar GUI, cujo objetivo é o projeto das interfaces gráficas do usuário.

Neste capítulo, o subfluxo "Projetar Camada de Apresentação" é detalhado, juntamente com suas atividades e artefatos. Além disso, um método é proposto para criação do Modelo Navegacional de uma aplicação Web, baseado nas extensões de UML propostas por J.Conallen [Conallen 99b] e N.Koch [Koch 01].

6.1 Visão Geral

O propósito do subfluxo "Projetar Camada de Apresentação" é atender aspectos não observados no fluxo de Análise e Projeto original e que são bastante relevantes em se tratando de aplicações Web. O primeiro aspecto atendido é o navegacional, que consiste na identificação de como o usuário caminha (navega) pela aplicação para utilizar as funcionalidades. O segundo aspecto atendido é o de apresentação, que consiste em como as interfaces gráficas do usuário são vistas (apresentadas).

Para tanto, faz-se necessária a criação do Modelo Navegacional, que possui como propósitos: orientar a navegação da aplicação, servir de base para o projeto das interfaces gráficas do usuário e identificar os elementos Web[9] necessários para criação da Camada de Apresentação no fluxo de Implementação.

O Modelo Navegacional é representado pelo diagrama de classes de UML com base nas extensões de UML propostas por J.Conallen [Conallen 99b] e N.Koch [Koch 01], e o processo de criação deste modelo é orientado por um método proposto, baseado na metodologia para o desenvolvimento de sistemas hipermídia proposta por H.Baumeister [Baumeister 99].

Desta forma, neste capítulo são apresentados: o detalhamento do subfluxo "Projetar Camada de Apresentação" e o método proposto para criação do Modelo Navegacional de uma aplicação Web.

6.2 O Subfluxo Projetar Camada de Apresentação

Este subfluxo tem como propósito estruturar a Camada de Apresentação de uma aplicação Web com base no Modelo Navegacional. Ao final da execução deste subfluxo, são alcançados os seguintes objetivos: a navegação da aplicação é definida, as interfaces gráficas do usuário são projetadas, e são identificados os elementos Web necessários para a criação da Camada de Apresentação da aplicação no fluxo de Implementação.

[9] Elementos específicos para aplicações Web, como página cliente, página servidor, scripts, entre outros.

A Figura 6-1 apresenta o detalhamento do subfluxo “Projetar Camada de Apresentação”, de forma que são identificadas as atividades realizadas e responsáveis, juntamente com os artefatos de entrada e os produzidos.

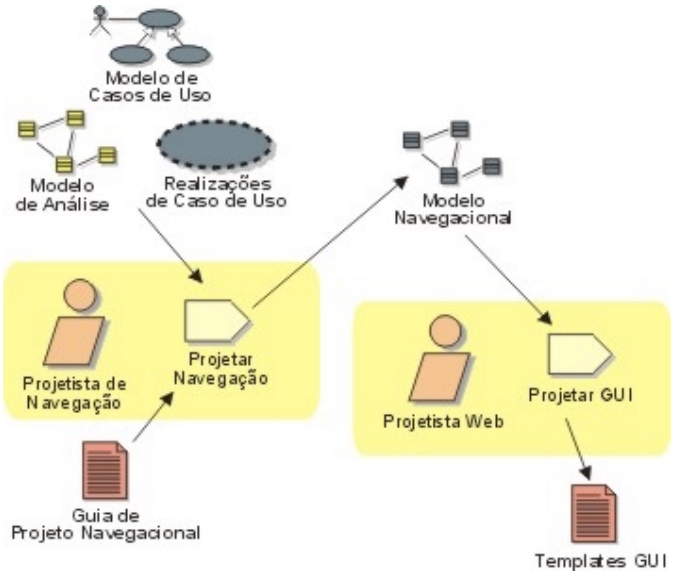


Figura 6-1 Detalhamento do Subfluxo Projetar Camada de Apresentação

A execução do subfluxo inicia com o processo de criação do Modelo Navegacional da aplicação, através da realização da atividade Projetar Navegação cujo responsável é o Projetista de Navegação. Para esta atividade ser realizada, são necessários como entrada os artefatos Modelo de Casos de Uso, Modelo de Análise e Realizações de Caso de Uso, que são produzidos em fases anteriores do processo de desenvolvimento; e o Guia de Projeto Navegacional, artefato criado para a orientar o projeto da navegação da aplicação.

O subfluxo termina com a realização da atividade Projetar GUI (*Graphic User Interface*) cujo responsável é o Projetista Web, com o objetivo de projetar as interfaces gráficas do usuário com base no Modelo Navegacional. Esta atividade produz o artefato Templates GUI que são esboços das interfaces gráficas do usuário, geralmente páginas HTML.

Nas subseções a seguir, as atividades deste subfluxo são detalhadas. Os propósitos das atividades são descritos e os artefatos necessários para realização de cada atividade são detalhados, juntamente com os artefatos produzidos.

6.2.1 A Atividade Projetar Navegação

Nesta atividade o Projetista de Navegação deve criar o Modelo Navegacional da aplicação Web, com os seguintes propósitos: identificar como o usuário caminha (navega) na aplicação para utilizar as funcionalidades, criar subsídios para o projeto das interfaces gráficas do usuário, e identificar os elementos Web necessários para a criação da Camada de Apresentação da aplicação no fluxo de Implementação.

Esta atividade utiliza como entrada alguns artefatos produzidos em fases anteriores do processo de desenvolvimento, são eles: o Modelo de Casos de Uso que descreve as funcionalidades que a aplicação Web fornece para os usuários, o Modelo de Análise que descreve as classes básicas do sistema com seus atributos e associações, e as Realizações de Caso de Uso que descrevem a colaboração entre objetos para os Casos de Uso. Além disso, foi criado o artefato Guia de Projeto Navegacional que fornece recomendações para o projeto da navegação de aplicações Web.

Nos parágrafos a seguir, são detalhados os artefatos que servem de entrada para a realização desta atividade, bem como o artefato produzido.

Modelo de Casos de Uso

O Modelo de Casos de Uso descreve as funcionalidades que a aplicação fornece para os usuários, e serve como contrato entre o cliente e os desenvolvedores. Inicialmente, este modelo contempla os requisitos funcionais do sistema, e serve como entrada essencial para a análise e projeto arquitetural. Na fase de Concepção, é usado para delimitar o escopo do sistema, e durante as fases de Elaboração e Construção, é refinado por fluxos de eventos mais detalhados. Este modelo deve ser mantido sempre consistente com o Modelo de Projeto.

O Analista de Sistemas é responsável pela integridade deste modelo, bem como pela garantia da sua corretude, consistência, e legibilidade. O Modelo de Casos de Uso é correto quando ele descreve a funcionalidade do sistema, e somente esta funcionalidade.

O Modelo de Casos de Uso é usado como entrada para realização da atividade Projetar Navegação, pois o Modelo Navegacional deve contemplar as navegações necessárias para satisfazer os Casos de Uso deste modelo.

Realizações de Caso de Uso

O artefato Realizações de Caso de Uso descreve como um particular Caso de Uso é realizado dentro do Modelo de Projeto, em termos de colaboração de objetos, é representado por diagramas de interação da UML. Na fase de Elaboração, as Realizações de Caso de Uso são criadas para os Casos de Uso relevantes para a arquitetura. Na fase de Construção, as Realizações de Caso de Uso são criadas para os Casos de Uso restantes.

O Projetista de Casos de Uso é responsável pela integridade das Realizações de Casos de Uso e deve garantir que elas reflitam o comportamento adequado e correto do Caso de Uso correspondente no Modelo de Casos de Uso, e que o fluxo de eventos do projeto esteja legível e que atenda seu propósito.

O artefato Realizações de Caso de Uso é usado como entrada para realização da atividade Projetar Navegação, pois o Modelo Navegacional deve contemplar as interações, entre o usuário e a aplicação, descritas neste artefato.

Modelo de Análise

O Modelo de Análise também chamado de Modelo Conceitual contém as Classes de Análise (básicas) da aplicação, seus atributos e associações. Este modelo é o resultado do processo de análise dos Casos de Uso e é uma abstração do Modelo de Projeto.

O Modelo de Análise é criado na fase de Elaboração e atualizado na fase de Construção. O Projetista é responsável pela integridade deste modelo, e deve garantir que sua manutenção no estado corrente reflita uma abstração do projeto.

O Modelo de Análise é usado como entrada para realização da atividade Projetar Navegação, pois o Modelo Navegacional é uma variação deste modelo, com ênfase nos aspectos de navegação e de apresentação da aplicação.

Guia de Projeto Navegacional

O artefato Guia de Projeto Navegacional, apresentado em detalhes no Apêndice A, define regras que orientam a construção do artefato Modelo Navegacional de aplicações Web.

Modelo Navegacional

O Modelo Navegacional descreve a navegação da aplicação, ou seja, como o usuário caminha (navega) na aplicação para utilizar as funcionalidades. Neste modelo são identificados os elementos Web que representam as interfaces gráficas da aplicação e os elementos Web responsáveis pela comunicação dessas interfaces gráficas com o sistema. Este modelo é representado pelo diagrama de classes utilizando extensões de UML.

O Modelo Navegacional serve de base para o projeto das interfaces gráficas da aplicação e é utilizado no fluxo de Implementação para a criação da Camada de Apresentação. O Projetista de Navegação é responsável por manter este modelo sempre consistente com o Modelo Conceitual da aplicação.

O Modelo Navegacional é produzido pela realização da atividade Projetar Navegação e seu processo de criação é orientado por um método que será apresentado na seção 6.3.

6.2.2 A Atividade Projetar GUI (Graphic User Interface)

Nesta atividade o Projetista Web deve projetar as interfaces gráficas da aplicação, com base no Modelo Navegacional. Esta atividade consiste em determinar a aparência dos elementos do Modelo Navegacional que representam as interfaces gráficas do usuário. Para garantir consistência, a aparência das interfaces gráficas do usuário deve obedecer ao padrão de apresentação (fonte, cor, botão, etc.) adotado pela organização. No parágrafo a seguir, é detalhado o artefato que é produzido por esta atividade.

Templates GUI

Este artefato apresenta (descreve) a aparência das interfaces gráficas do usuário. Geralmente, são páginas HTML com a representação gráfica dos elementos definidos no Modelo Navegacional, que representam as páginas cliente e formulários da aplicação Web.

6.3 O Método para Criação do Modelo Navegacional

O método proposto para criação do Modelo Navegacional de uma aplicação Web utiliza conceitos, tais como primitivas de acesso, apresentados na extensão de UML proposta por N.Koch [Koch 01], que por sua vez baseia-se no OOHDM. Para representar os elementos do Modelo Navegacional, são utilizados estereótipos definidos na extensão de UML proposta por J.Connalen [Conallen 99b] pois são específicos para o caso de aplicações Web. Para identificação de caminhos navegacionais, o método proposto baseia-se na metodologia para desenvolvimento de sistemas hipermídia apresentada em [Baumeister 99]. O método proposto define passos para criação do Modelo Navegacional, além de padrões de projeto navegacional para satisfazer operações de manutenção de objetos do sistema.

O método contempla a modelagem da navegação de aplicações Web que possam ter funcionalidades como as de qualquer outro tipo de aplicação tradicional. Os passos necessários para criação do Modelo Navegacional são os seguintes: Identificar as Classes Navegacionais, Identificar os Caminhos Navegacionais e Criar os Caminhos Navegacionais. Nas subseções a seguir, estes passos são detalhados.

6.3.1 Primeiro Passo: Identificar as Classes Navegacionais

O processo de criação do Modelo Navegacional inicia com a identificação das classes do Modelo de Análise que são relevantes em termos de navegação. A identificação destas classes, chamadas de classes navegacionais, é orientada pelo Modelo de Casos de Uso e pelas Realizações de Casos de Uso. O Projetista Navegacional deve considerar apenas as classes básicas do Modelo de Análise que necessitem de navegação; as classes básicas que apenas são referenciadas no Modelo de Análise são reduzidas a atributos das classes que as referenciam, no Modelo Navegacional.

As classes navegacionais são representadas no Modelo Navegacional fazendo-se referência ao mesmo nome da classe básica correspondente no Modelo de Análise. A Figura 6-2 apresenta o padrão de projeto navegacional para representar as classes navegacionais, no Modelo Navegacional de uma aplicação Web. O lado esquerdo da Figura 6-2 apresenta o padrão de projeto navegacional, utilizando estereótipos da extensão de UML proposta por J.Conallen [Conallen 99b], já o lado direito apresenta o mesmo padrão, porém, utilizando a representação gráfica dos estereótipos.

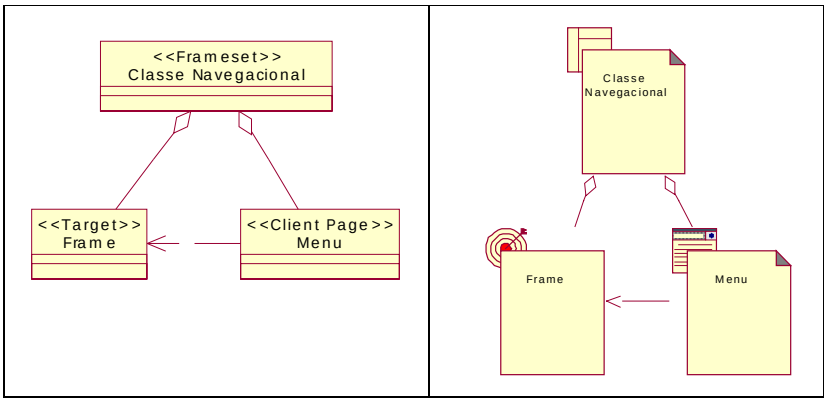


Figura 6-2 Padrão de Projeto Navegacional para Classes Navegacionais

A seguir, é descrito a função (papel) de cada classe do padrão de projeto navegacional para classes navegacionais de uma aplicação Web, apresentado na Figura 6-2.

- A classe chamada Classe Navegacional representa o fato de que, no *browser*, as interfaces gráficas do usuário são exibidas através de páginas cliente com mais de um frame.

- A classe Frame representa o frame principal das páginas cliente, no qual são montadas as interfaces gráficas do usuário, exibidas nos caminhos navegacionais que satisfazem as funcionalidades oferecidas pela classe navegacional.

- A classe Menu representa a página cliente responsável por exibir ao usuário as funcionalidades oferecidas pela Classe Navegacional, com seus respectivos links.

6.3.2 Segundo Passo: Identificar os Caminhos Navegacionais

Os caminhos navegacionais representam como o usuário caminha (navega) na aplicação para utilizar as funcionalidades do sistema. Eles são identificados através das associações entre as classes básicas do Modelo de Análise. O Modelo de Casos de Uso é utilizado pelo Projetista de Navegação para verificar se todas as funcionalidades do sistema estão contempladas na navegação, e as Realizações de Caso de Uso são usadas para verificar se a seqüência de eventos para a realização (execução) do Caso de Uso está contemplada e é obedecida pelo caminho navegacional.

O sentido dos caminhos navegacionais entre classes navegacionais é determinado pela direção da associação entre as classes básicas correspondentes no Modelo de Análise, entretanto, este sentido pode ser bidirecional. Para cada Classe Navegacional, são relevantes apenas as associações que a têm como origem e para cada uma dessas associações, um caminho navegacional é criado entre as classes navegacionais origem e destino, dependendo da multiplicidade na origem da associação.

A Figura 6-3 apresenta um exemplo de parte de um Modelo de Análise com classes básicas e associações. Este Modelo de Análise mostra exemplos de associações com os dois tipos básicos de multiplicidade (igual a um e maior que um) na origem, que influenciam o aspecto navegacional de uma aplicação, ou seja, originam caminhos navegacionais distintos.

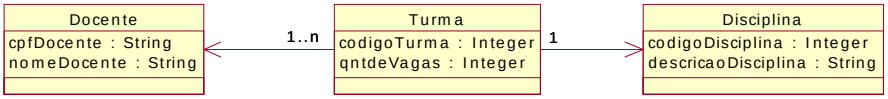


Figura 6-3 Exemplo de parte de um Modelo de Análise

A seguir, são identificados os caminhos navegacionais observados para o Modelo de Análise apresentado na Figura 6-3.

Um caminho navegacional é originado da associação entre as classes Turma e Disciplina. Esta associação possui multiplicidade igual a um na origem, o que significa que um objeto da classe Turma possui como atributo um objeto da classe Disciplina. Diante deste panorama, são obtidas as seguintes conclusões:

- A interface gráfica responsável pela operação de inclusão de objetos da classe Turma deve possuir um campo para identificar o objeto da classe Disciplina. Este campo pode ser representado por um campo simples para entrada, uma lista, ou um link para uma funcionalidade que auxilie a busca de objetos da classe Disciplina.

- A interface gráfica responsável pela operação de detalhamento dos objetos da classe Turma deve possuir um campo que identifique o objeto contido da classe Disciplina, com a opção de detalhamento através de link.

Um caminho navegacional é originado da associação entre as classes Turma e Docente. Esta associação possui multiplicidade maior que um na origem, o que significa que um objeto da classe Turma possui como atributo uma coleção de objetos da classe Docente. Diante deste panorama, são obtidas as seguintes conclusões:

- Deve ser criada uma funcionalidade (oferecida pela classe Turma) cuja interface gráfica possa manipular (incluir, alterar, excluir) a coleção de objetos da classe Docente que pertencem a um objeto da classe Turma.

- A interface gráfica responsável pela operação de detalhamento dos objetos da classe Turma deve possuir um campo do tipo link, que quando acionado faça referência a uma funcionalidade cujo propósito seja o de exibir através de uma estrutura de índice, a coleção de objetos da classe Docente que pertencem a um objeto da classe Turma. Dependendo do tamanho da coleção de objetos, pode-se ter uma funcionalidade intermediária que auxilie a busca dos objetos da coleção.

Pode-se ter outras estratégias (tipos) de implementação para os caminhos navegacionais derivados das associações entre as classes básicas do Modelo de Análise apresentado na Figura 6-3. Além dos caminhos navegacionais originados destas associações, outros caminhos navegacionais são necessários para satisfazer as operações de manutenção (inclusão, alteração, exclusão e consulta) dos objetos das classes do sistema. Estes caminhos navegacionais são detalhados na próxima subseção (6.3.3).

6.3.3 Terceiro Passo: Criar os Caminhos Navegacionais

O Modelo Navegacional contempla os caminhos navegacionais para satisfazer todas as funcionalidades oferecidas pelas classes navegacionais. Estes caminhos navegacionais são resultantes das associações entre as classes básicas do Modelo de Análise conforme apresentado na subseção 6.3.2 e, são criados para satisfazer as operações de manutenção dos objetos das classes do sistema.

Os caminhos navegacionais que satisfazem as operações de manutenção dos objetos das classes do sistema seguem padrões propostos de projeto navegacional, baseados no padrão arquitetural *Thick Web Client*[10], proposto por J.Connalen [Conallen 99b]. Nos parágrafos a seguir, esses padrões propostos de projeto navegacional são detalhados.

Padrão de Projeto Navegacional para operação de Inclusão

A Figura 6-4 apresenta o padrão de projeto para a modelagem navegacional da operação de inclusão dos objetos do sistema.

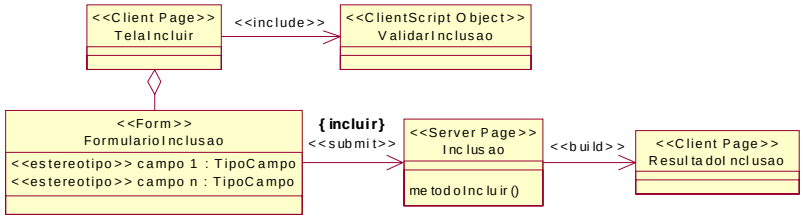


Figura 6-4 Padrão de Modelagem Navegacional para operação de Inclusão

A seguir, é detalhado a função (papel) de cada classe e associação do padrão de projeto navegacional apresentado na Figura 6-4.

- A classe TelaIncluir representa a página cliente responsável por exibir a interface gráfica do usuário para inclusão de objetos.

- A classe FormularioInclusao representa o formulário contendo os campos a serem informados pelo usuário. Os campos são originados dos atributos das classes básicas e são representados através de atributos estereotipados (ver Anexo A) na classe FormularioInclusao. O estereótipo de cada atributo (*input*, *select*, *button*, *link*, entre outros) especifica como cada campo é apresentado na interface gráfica.

- A associação entre a classe FormularioInclusao e a classe Inclusao representa a ação de confirmação por parte do usuário ({incluir}) para a inclusão do objeto no meio de persistência (geralmente, banco de dados) da aplicação.

[10] Padrão arquitetural para aplicações Web que assume processamento no *browser* do usuário, ou seja, é usado em sistemas que utilizam scripts, *applets* ou controles *ActiveX* no lado cliente.

- A classe ValidarInclusao representa os scripts executados no cliente, responsáveis pela validação das informações preenchidas pelo usuário.

- A classe Inclusao representa a página servidor responsável pela validação no servidor das informações vindas da interface gráfica, bem como pela execução da operação de inclusão do objeto no meio físico de persistência da aplicação. O método que é chamado (acionado) para de fato realizar a operação de inclusão é representado pela operação metodoIncluir da classe Inclusao.

- A associação entre a classe Inclusao e a classe ResultadoInclusao representa a construção do resultado do processamento no servidor no *browser* do usuário, através de uma página cliente representada pela classe ResultadoInclusao.

Padrão de Projeto Navegacional para as demais operações de Manutenção

Este padrão de projeto navegacional contempla as operações de pesquisa, alteração e exclusão de objetos das classes básicas na aplicação e, é apresentado na Figura 6-5.

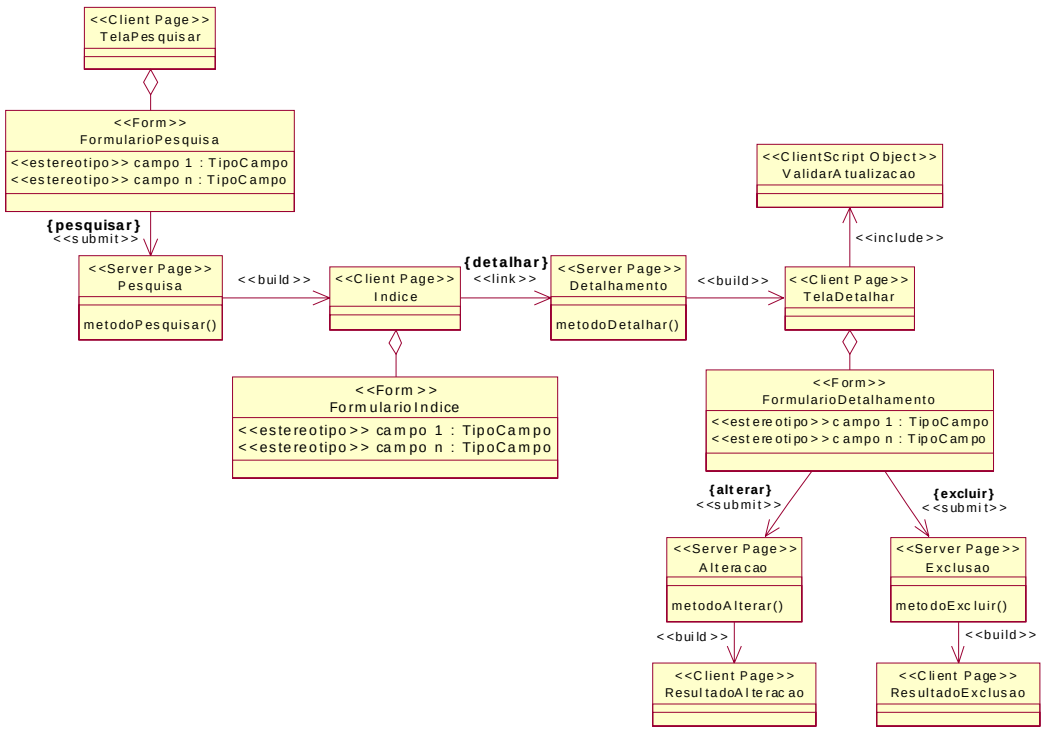


Figura 6-5 Padrão de Modelagem Navegacional para operações de Pesquisa, Alteração e Exclusão

A seguir, é detalhada a função (papel) de cada classe e associação do padrão de projeto navegacional apresentado na Figura 6-5.

- A classe TelaPesquisar representa a página cliente responsável por exibir a interface gráfica do usuário para auxiliar a pesquisa de objetos.

- A classe FormularioPesquisa representa o formulário contendo um ou mais campos a serem informados pelo usuário para auxiliar a pesquisa de objetos no meio de persistência da aplicação. Os campos são representados através de atributos estereotipados na classe FormularioPesquisa. O estereótipo de cada atributo especifica como cada campo é apresentado na interface gráfica.

- A associação entre a classe FormularioPesquisa e a classe Pesquisa, representa a ação de confirmação por parte do usuário ({pesquisar}) para pesquisa de objetos no meio físico de persistência da aplicação, de acordo com os campos (parâmetros) informados.

- A classe Pesquisa representa a página servidor responsável pelo processamento no servidor, da operação de pesquisa de objetos de acordo com os parâmetros vindos da interface gráfica de pesquisa. O método que é chamado (acionado) para de fato realizar a operação de pesquisa é representado pela operação metodoPesquisar da classe Pesquisa.

- A associação entre a classe Pesquisa e a classe Indice representa a construção no *browser* do usuário, da coleção de objetos retornados do processamento da operação de pesquisa, organizados por uma estrutura de índice, através de uma página cliente representada pela classe Indice. A identificação dos objetos é feita por seus atributos mais significativos, que são representados através de atributos estereotipados da classe FormularioIndice.

- A associação entre a classe Indice e a classe Detalhamento representa a ação por parte do usuário ({detalhar}) de selecionar um dos objetos da coleção para a apresentação do seu detalhe.

- A classe Detalhamento representa a página servidor responsável pelo processamento no servidor, da operação de detalhar o objeto selecionado pelo usuário. O método que é chamado (acionado) para de fato realizar a operação de detalhamento do objeto é representado pela operação metodoDetalhar da classe Detalhamento.

- A associação entre a classe Detalhamento e a classe TelaAtualizar representa a construção do detalhamento do objeto no *browser* do usuário.

- A classe TelaDetalhar representa a página cliente responsável por exibir o detalhamento do objeto, e posteriores operações de alteração e exclusão.

- A classe FormularioDetalhamento representa o formulário contendo a coleção de campos preenchidos com as informações do objeto. Os campos são representados através de atributos estereotipados na classe FormularioDetalhamento. O estereótipo de cada atributo especifica como cada campo é apresentado na interface gráfica.

- A classe ValidarAtualizacao representa os scripts executados no cliente, responsáveis pela validação das informações do formulário de detalhamento quando acionada a operação de alteração.

- A associação entre a classe FormularioDetalhamento e a classe Alteracao representa a ação de confirmação por parte do usuário ({alterar}) para alteração do objeto no meio físico de persistência da aplicação.

- A classe Alteracao representa a página servidor responsável pela validação no servidor, das informações vindas da interface gráfica, bem como pela execução da operação de alteração do objeto no meio de persistência da aplicação. O método que é chamado (acionado) para de fato realizar a operação de alteração é representado pela operação metodoAlterar da classe Alteracao.

- A associação entre a classe Alteracao e a classe ResultadoAlteracao representa a construção do resultado do processamento no servidor no *browser* do usuário, através de uma página cliente representada pela classe ResultadoAlteracao.

- A associação entre a classe FormularioDetalhamento e a classe Exclusao representa a ação de confirmação por parte do usuário ({excluir}) para exclusão do objeto no meio físico de persistência da aplicação.

- A classe Exclusao representa a página servidor responsável pela validação no servidor, das informações vindas da interface gráfica, bem como pela execução da operação de exclusão do objeto no meio físico de persistência da aplicação. O método que é chamado (acionado) para de fato realizar a operação de exclusão é representado pela operação metodoExcluir da classe Exclusao.

- A associação entre a classe Exclusao e a classe ResultadoExclusao representa a construção do resultado do processamento no servidor no *browser* do usuário, através de uma página cliente representada pela classe ResultadoExclusao.

A operação de pesquisa pode ser simplificada dependendo do contexto, pois se a pesquisa for feita unicamente por um campo chave, o processo deve considerar apenas o retorno de um objeto, desconsiderando a estrutura de índices. Variações ou adaptações na modelagem navegacional para as operações de manutenção podem ser facilmente realizadas.

Após a identificação dos caminhos navegacionais e das considerações apresentadas na subseção 6.3.2, bem como a apresentação dos padrões de projeto navegacional para operações de manutenção (inclusão, alteração, pesquisa e exclusão), a Figura 6-6 apresenta o Modelo Navegacional derivado do Modelo de Análise apresentado na Figura 6-3.

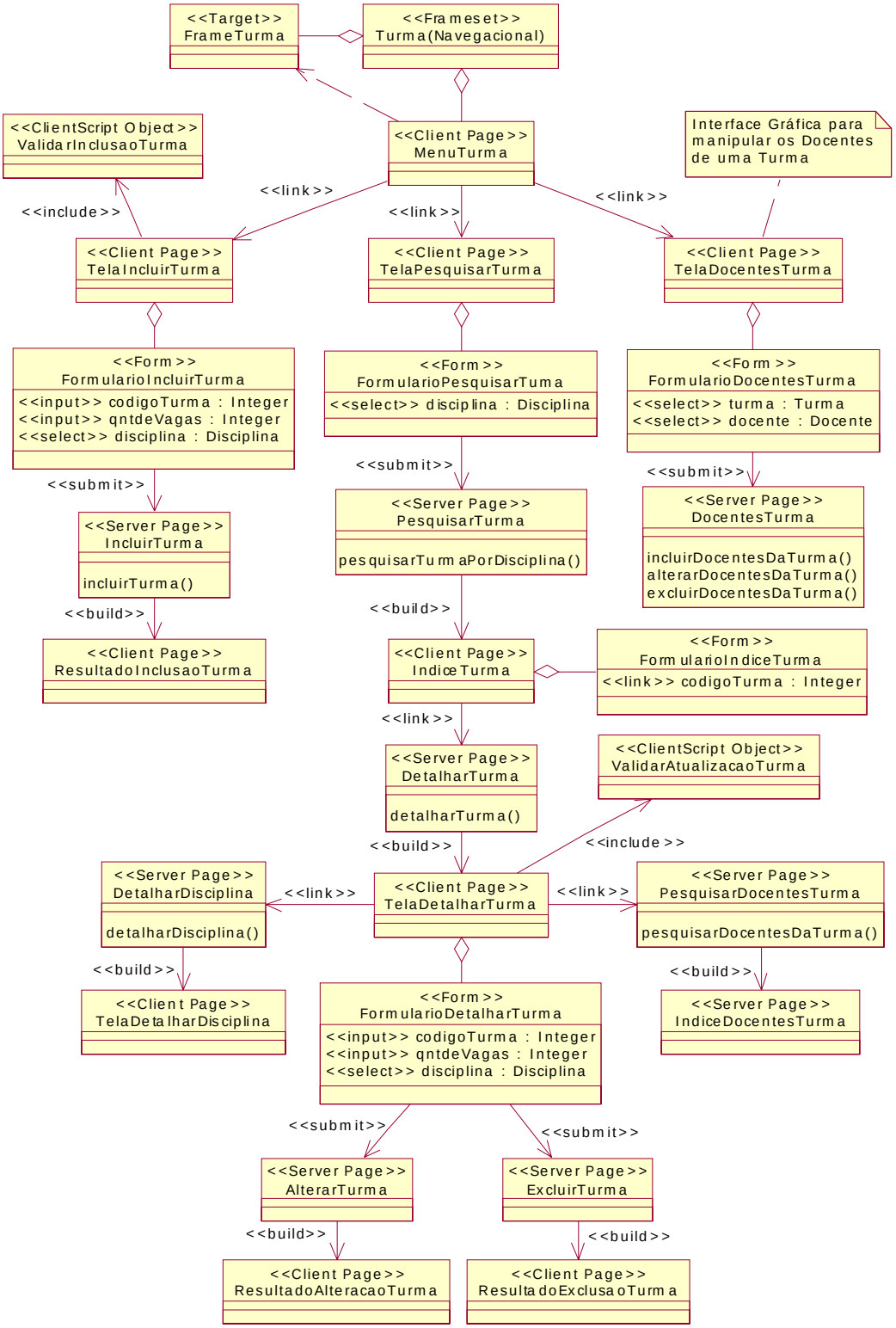


Figura 6-6 Exemplo de Modelo Navegacional

6.4 Considerações Finais

Neste capítulo, foi apresentado o detalhamento do subfluxo “Projetar Camada de Apresentação”, cujo propósito é atender aos aspectos de navegação e de apresentação, que são de grande relevância quando se trata de aplicações Web. Ao final da execução do subfluxo, os desenvolvedores possuem os subsídios necessários para criação da Camada de Apresentação da aplicação, no fluxo de Implementação.

Neste capítulo, foi também apresentado um método proposto para a criação do Modelo Navegacional de uma aplicação Web, baseado na metodologia para desenvolvimento de sistemas hipermídia proposta por H.Baumeister [Baumeister 99], e em extensões de UML propostas por J.Connalen [Conallen 99b] e N.Koch [Koch 01].

O próximo capítulo apresenta a validação do subfluxo proposto, através de sua aplicação a um estudo de caso realizado sobre a aplicação Web SIG@UFPE (Sistema de Informações e Gestão Acadêmica da UFPE).

Capítulo 7

Estudo de Caso – A Aplicação Web SIG@UFPE

A extensão do fluxo de Análise e Projeto do RUP para o desenvolvimento mais apropriado de aplicações Web explorou as características da metodologia de adequabilidade a diferentes tipos de aplicação e generecidade do processo, e consistiu na adaptação de atividades já existentes e na criação de um novo subfluxo para atender aspectos não observados originalmente, relacionados à criação da Camada de Apresentação da aplicação. Para tanto, novas atividades e artefatos foram criados e, portanto, devem ser validados através de um estudo de caso, com o propósito de verificar suas adequações no processo e seus benefícios.

Neste capítulo, um estudo de caso é apresentado com ênfase no detalhamento da execução do subfluxo "Projetar Camada de Apresentação". Para isto, os artefatos que servem como base para realização das atividades deste subfluxo, produzidos em fases anteriores do fluxo de Análise e Projeto, são apresentados, bem como um resumo sobre as características da aplicação.

A aplicação Web utilizada como estudo de caso é o SIG@UFPE (Sistema de Informação e Gestão Acadêmica da UFPE), desenvolvido no padrão arquitetural de camadas, utilizando Java como linguagem de programação e JSP/*Servlets* na Camada de Apresentação. Esta aplicação está sendo desenvolvida sob a gerência e coordenação do Núcleo de Tecnologia da Informação desta universidade.

7.1 Visão Geral

A extensão do fluxo de Análise e Projeto do RUP para o desenvolvimento de aplicações Web consistiu na criação de um novo subfluxo para atender aspectos relacionados à criação da Camada de Apresentação da aplicação, tendo como base o artefato Modelo Navegacional. Para tanto, surgiu a necessidade de orientações para a criação deste artefato. Desta forma, na seção 6.3, um método para criação do Modelo Navegacional foi proposto, baseado nas extensões de UML propostas por J.Conallen [Conallen 99b] e N.Koch [Koch 01].

Além da criação de um novo subfluxo, atividades já existentes do fluxo de Análise e Projeto foram adaptadas (estendidas) para satisfazer, mais apropriadamente, as necessidades do desenvolvimento de aplicações Web, com base no framework de Análise e Projeto proposto por A.Araújo [Araújo 01].

Diante deste panorama, o estudo de caso tem como objetivo validar a viabilidade da proposta de extensão do fluxo de Análise e Projeto. Entretanto, parte desta extensão baseou-se e já foi validada no framework de Análise e Projeto proposto por A.Araújo [Araújo 01]. Desta forma, o estudo de caso apresentado neste capítulo tem como foco a execução do novo subfluxo e do método proposto para a criação do Modelo Navegacional da aplicação.

7.2 Preparação do Ambiente

Os artefatos apresentados neste estudo de caso são modelos de UML produzidos através da ferramenta de modelagem de sistemas Rational Rose (versão 2001.03.00). Esta ferramenta precisou ser configurada para incorporar as extensões de UML utilizadas no Modelo Navegacional, bem como o método proposto para criação deste artefato.

No caso das extensões de UML, foram feitas configurações na ferramenta Rational Rose para que os componentes do diagrama de classes (classes, associações e atributos) pudessem ser estereotipados (inclusive com a representação gráfica) para expressar os elementos do Modelo Navegacional.

A ferramenta Rational Rose foi também configurada para incorporar uma nova funcionalidade – a criação do Modelo Navegacional. Para tanto, foi elaborado um programa escrito em uma linguagem baseada em scripts, que manipula uma API (*Application Program Interface*) denominada REI (*Rational Extensibility Interface*), fornecida pela ferramenta. Este programa baseia-se no método proposto para criação do Modelo Navegacional (seção 6.3). Desta forma, é solicitado como entrada o Modelo de Análise, e baseado nos componentes deste modelo (classes e seus atributos, associações e suas multiplicidades) é criada uma nova visão com ênfase nos aspectos de navegação e de apresentação – o Modelo Navegacional, que, posteriormente, necessita de um refinamento por parte do projetista da aplicação.

Na seção a seguir, a aplicação que serviu de estudo de caso é detalhada.

7.3 A Aplicação Web SIG@UFPE

A informação é fator primordial para o planejamento e execução de ações com o propósito de melhorar os serviços oferecidos por uma instituição de ensino superior. Desta forma, faz-se necessária, a criação de um sistema de informação robusto que possibilite, entre outras coisas, a descentralização e maior agilidade dos processos na instituição de ensino. Diante deste panorama, a aplicação Web SIG@UFPE visa atender, num primeiro momento, o principal serviço oferecido pela universidade, que é o ensino em todos os níveis, ou seja, graduação, pós-graduação e extensão.

7.3.1 Objetivos da Aplicação

O sistema de informação e gestão acadêmica da UFPE está sendo desenvolvido em módulos que correspondem a etapas (fases) do ano letivo, ou seja, planejamento de matrícula, matrícula, acompanhamento e encerramento de período, entre outras funcionalidades, como integralização curricular, conclusão de curso e emissão de documentos oficiais como diplomas e certificados.

Inicialmente, seguindo como prioridade a ordem cronológica das etapas, foi desenvolvido o módulo do sistema que abrange as fases de planejamento de matrícula e matrícula. Para tanto, foi também necessário o desenvolvimento de subsistemas, basicamente de cadastros, com o propósito de controlar aspectos relacionados à estrutura (espaço) física e organizacional da instituição, bem como os recursos humanos.

Portanto, o estudo de caso tem como foco algumas funcionalidades do módulo de planejamento de matrícula e matrícula. Este módulo possui, como principais objetivos, auxiliar e controlar o planejamento da matrícula com a finalidade de criar a infra-estrutura necessária para a matrícula, bem como possibilitar a matrícula descentralizada, podendo o próprio discente realizar sua matrícula através da Internet.

7.3.2 Usuários da Aplicação

Os usuários da aplicação Web SIG@UFPE são toda a comunidade acadêmica da universidade, mais os servidores responsáveis pela operacionalização das ações que dizem respeito ao ensino nos níveis de graduação, pós-graduação e extensão. Diante deste panorama, um grande número de perfis funcionais do sistema é identificado, correspondendo às diferentes funções que são relevantes no contexto do ensino.

Para o módulo de planejamento de matrícula e matrícula que está sendo usado como base para o estudo de caso, os principais perfis funcionais são os seguintes:

- Docente – responsável por ministrar atividades acadêmicas em que são especialistas, nos três níveis de ensino da universidade.
- Discente – possui vínculo em um ou mais cursos oferecidos pela universidade; cada vínculo é obtido através de algum meio de seleção, dependendo do nível de ensino.
- Técnico Administrativo – responsável pela operacionalização e controle das atividades administrativas referentes ao ensino.
- Coordenador – docente responsável pela coordenação de um curso ou área de ensino da universidade.
- Chefe de Departamento – docente responsável pela gerência (chefia) de um departamento (órgão gerenciador de um ou mais cursos).

Observando os tipos de perfis funcionais da aplicação descritos acima, nota-se que um mesmo usuário pode ter mais de um perfil funcional no sistema, ou seja, visibilidades diferentes com relação às funcionalidades. Desta forma, foi necessário o desenvolvimento de um subsistema de controle de acesso robusto o suficiente, para atender esta e outros tipos de situação, no que diz respeito ao acesso às funcionalidades da aplicação.

7.3.3 Principais Funções da Aplicação

A aplicação Web SIG@UFPE possui como macro funções, realizar (atender) as etapas (fases) do ano letivo da universidade. Como o foco do estudo de caso é o módulo que controla as fases de planejamento de matrícula e matrícula, são descritas nos parágrafos a seguir, algumas das principais funções deste módulo.

Especificação das Atividades Acadêmicas a Ofertar

Consiste em o coordenador e o chefe de departamento especificarem um conjunto de atividades acadêmicas (componente curricular tal como disciplina eletiva, disciplina obrigatória e estágio) a ofertar no período, baseados na obrigatoriedade curricular e em informações estatísticas de períodos anteriores.

Intenção de Matrícula

Também chamada de pré-matrícula, consiste em o discente manifestar a intenção de cursar uma ou mais atividades acadêmicas do tipo eletivas, ofertadas pelo departamento responsável pelo curso para o período.

Definição das Atividades Acadêmicas Ofertadas

Consiste em o coordenador e o chefe de departamento definirem de forma definitiva as atividades acadêmicas ofertadas para o período, com base nas estatísticas da intenção de matrícula dos discentes.

Criação de Turmas e Subturmas

Consiste em o coordenador ou técnico administrativo criarem as turmas para o período, especificando a atividade acadêmica, espaço físico (salas), horários nos dias da semana, quantidade de vagas e professor(es) ministrante(s). Para alguns casos, como por exemplo, quantidade de vagas muito grande e limitação do espaço físico, algumas turmas são divididas em subturmas independentes uma das outras.

Matrícula em Atividade Acadêmica

Consiste em o discente realizar matrícula em atividades acadêmicas que não originam a criação de turma, como estágios e congressos, ou seja, não necessitam de espaço físico e horário permanente.

Matrícula em Turma

Consiste em o discente realizar matrícula em turmas criadas para o período, sendo que esta ação pode estar sujeita a análise de prioridade de acordo com a situação do discente, caso haja maior procura que a oferta.

7.4 A Execução do Subfluxo Projetar Camada de Apresentação

A funcionalidade "Criação de Turmas e Subturmas" foi eleita para ser detalhada levando em consideração suas características e grau de complexidade, que se mostram adequados para a exemplificação da execução do subfluxo "Projetar Camada de Apresentação".

A execução deste subfluxo consiste na realização das atividades Projetar Navegação e Projetar GUI. Para tanto, alguns artefatos que são produzidos em fases anteriores do processo de desenvolvimento são necessários. Na próxima subseção, estes artefatos são apresentados.

7.4.1 Artefatos Base para Execução do Subfluxo

Nos parágrafos a seguir, são apresentados os artefatos Modelo de Casos de Uso, Realizações de Caso de Uso e Modelo de Análise ou Conceitual, para contemplar (satisfazer) a funcionalidade "Criação de Turmas e Subturmas" da aplicação Web; e que servem de entrada (base) para criação do Modelo Navegacional, que por sua vez é o responsável pela estruturação da Camada de Apresentação.

Modelo de Casos de Uso

O Modelo de Casos de Uso é usado como entrada essencial para as atividades de análise, projeto e testes, e é também bastante relevante para a criação do Modelo Navegacional, pois é usado para verificar se todos os Casos de Uso são satisfeitos e estão contemplados na Camada de Apresentação que é a responsável pela interação entre o usuário e a aplicação.

A Figura 7-1 apresenta o Modelo de Casos de Uso usado como ponto de partida para o desenvolvimento do estudo de caso para o projeto da Camada de Apresentação. Neste modelo, os Casos de Uso relevantes para o entendimento da pequena parte da aplicação Web usada no estudo de caso são identificados, juntamente com os atores que interagem com as funcionalidades da aplicação representadas por esses Casos de Uso.

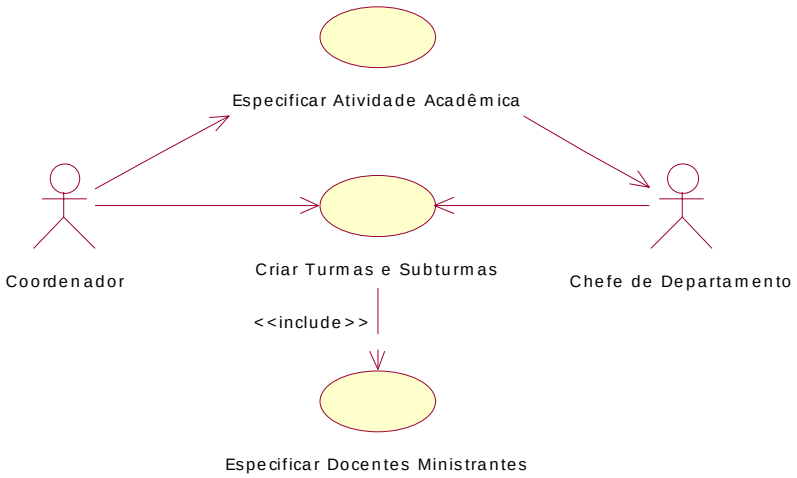


Figura 7-1 Modelo de Casos de Uso da funcionalidade "Criação de Turmas e Subturmas"

A seguir, são descritos os Caso de Uso do modelo apresentado na Figura 7-1.

- **Especificar Atividade Acadêmica:** o coordenador especifica (define) os componentes curriculares que serão ofertados no período letivo através de uma atividade acadêmica, baseado na obrigatoriedade curricular e em informações estatísticas do processo de intenção de matrícula (pré-matrícula) dos discentes.

- **Criar Turmas e Subturmas:** o coordenador e/ou chefe de departamento cria turmas para as atividades acadêmicas que necessitam de espaço físico (salas), e horários permanentes nos dias da semana. A turma pertence a um turno e é de um tipo (fixa ou variável com relação ao número de discentes). Para cada turma é definido: a natureza (teórica ou prática), o número de aulas, carga horária e quantidade de vagas. A turma é dividida em subturmas perante algumas situações como divisão para aulas práticas, grande demanda de alunos e limitação do espaço físico. A turma que possui subturmas pode ter alunos matriculados ou não, no segundo caso a turma é chamada de agregadora.

- **Especificar Docentes Ministrantes:** o coordenador e/ou chefe de departamento especifica o(s) docente(s) responsável(eis) por ministrar a atividade acadêmica para uma turma, juntamente com a carga horária. O docente é candidato a ministrar a atividade acadêmica dependendo da sua habilitação (especialização) no conteúdo do componente curricular vinculado.

Realizações de Caso de Uso

O artefato Realizações de Caso de Uso descreve o comportamento dinâmico de um Caso de Uso, ou seja, descreve como um particular Caso de Uso é realizado dentro do Modelo de Projeto, em termos de colaboração de objetos. O Modelo Navegacional deve contemplar (satisfazer) as interações do usuário com a aplicação, descritas neste artefato.

A Figura 7-2 apresenta as Realizações do Caso de Uso “Criar Turmas e Subturmas”, representado por um diagrama de seqüência de UML que, conceitualmente, enfatiza a ordenação no tempo das mensagens. Para simplificar o entendimento, devido à relação de *include* dos Casos de Uso do modelo apresentado na Figura 7-1, as Realizações do Caso de Uso “Especificar Docentes Ministrantes” também estão contempladas neste diagrama de seqüência.

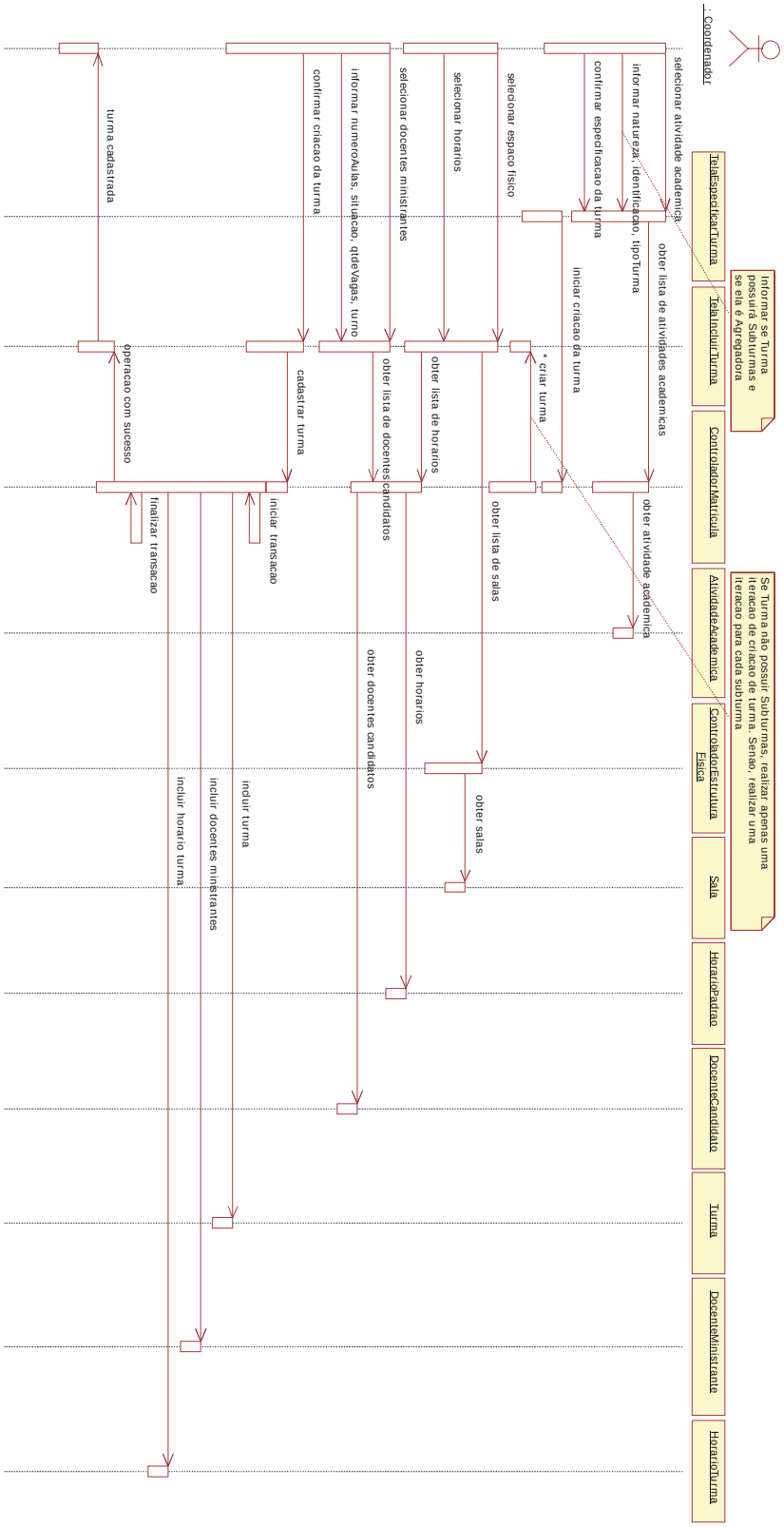


Figura 7-2 Realizações do Caso de Uso “Criar Turmas e Subturmas”

Modelo de Análise (ou Conceitual)

O Modelo de Análise é o resultado da análise dos Casos de Uso e de suas realizações. Neste modelo são identificadas as classes básicas da aplicação e suas associações. O Modelo Navegacional é uma variação do Modelo Conceitual, com ênfase nos aspectos de navegação e de apresentação da aplicação.

A Figura 7-3 apresenta o Modelo de Análise representado por um diagrama de classes de UML, resultante da análise dos Casos de Uso do modelo apresentado na Figura 7-1.

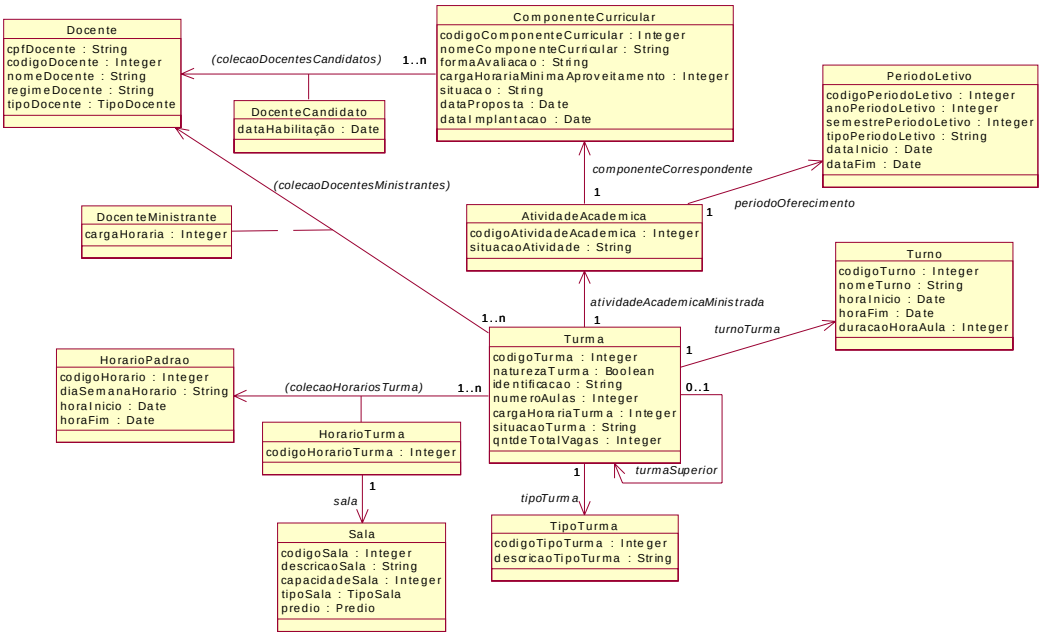


Figura 7-3 Modelo de Análise ou Conceitual da funcionalidade "Criação de Turmas e Subturmas"

7.4.2 Realização da Atividade Projetar Navegação

O propósito desta atividade é criar subsídios para criação da camada de apresentação no fluxo de Implementação e para o projeto das interfaces gráficas da aplicação. Para tanto, faz-se necessária a criação do Modelo Navegacional, baseado nos artefatos apresentados na subseção 7.4.1. A Figura 7-4 apresenta o Modelo Navegacional da funcionalidade "Criação de Turmas e Subturmas", gerado automaticamente através da rotina integrada ao ambiente da ferramenta de modelagem Rational Rose, conforme descrito na seção 7.2.

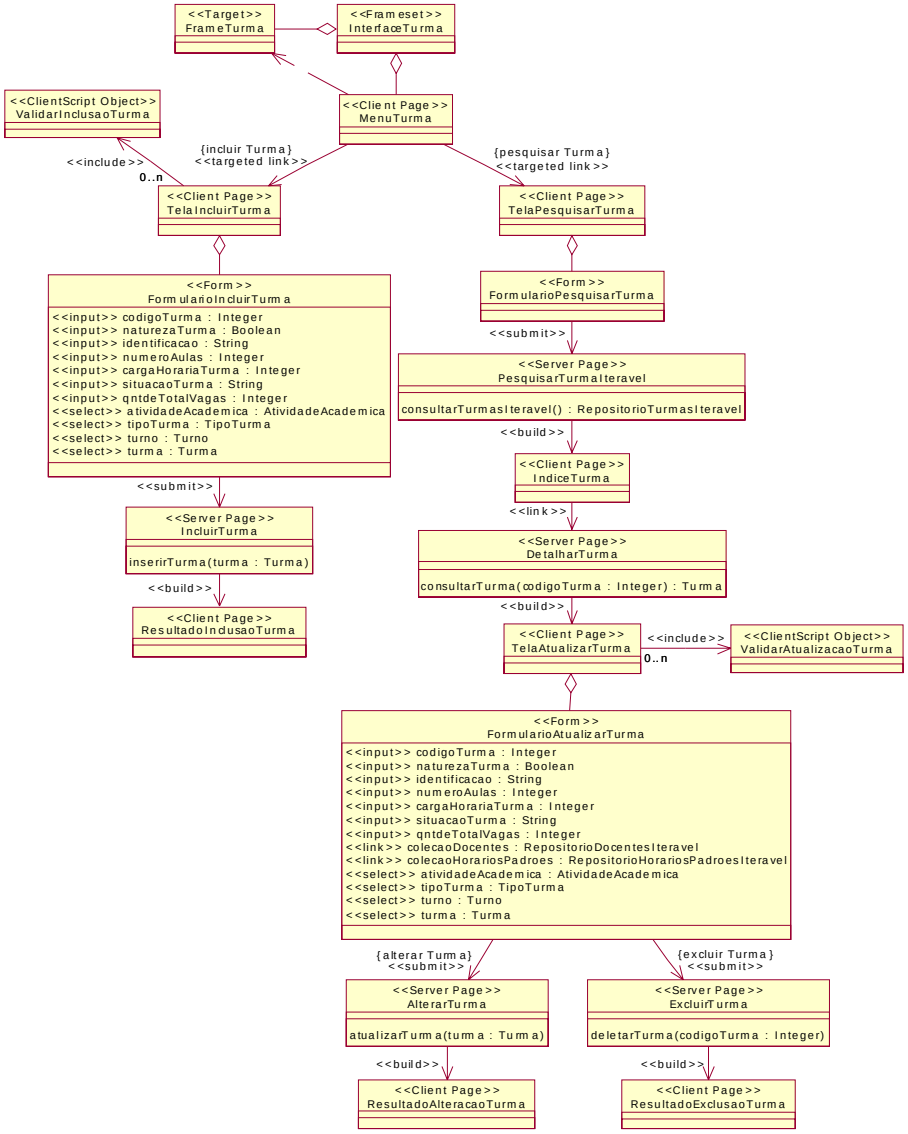


Figura 7-4 Exemplo de Modelo Navegacional gerado automaticamente

O Modelo Navegacional apresentado na Figura 7-4 precisa ser refinado e talvez modificado pelo projetista de navegação para melhor atender as necessidades de navegação da funcionalidade do sistema. Desta forma, a Figura 7-5 apresenta parte do Modelo Navegacional refinado da funcionalidade “Criação de Turmas e Subturmas”.

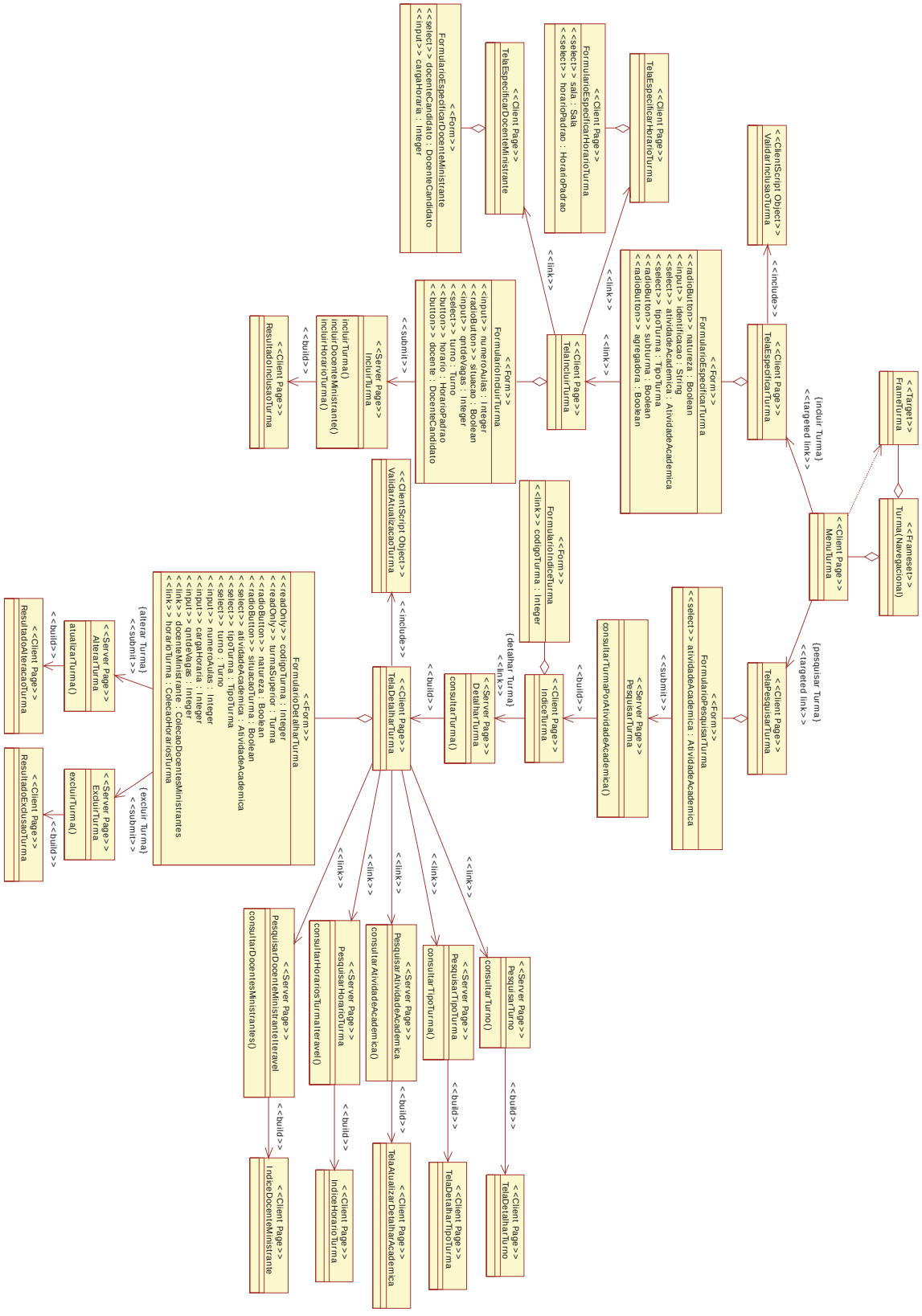


Figura 7-5 Modelo Navegacional da funcionalidade “Criação de Turmas e Subturmas”

7.4.3 Realização da Atividade Projetar GUI

O propósito desta atividade é o projeto da apresentação dos elementos do Modelo Navegacional (classes que representam páginas cliente e formulários) que representam as interfaces gráficas para interação do usuário com a aplicação. Desta forma, são criados *templates* (esboços) dessas interfaces gráficas, geralmente, páginas HTML.

A Figura 7-6 apresenta a interface gráfica para especificar uma turma, já integrada à aplicação Web SIG@UFPE, derivada da classe *FormularioEspecificarTurma* do Modelo Navegacional apresentado na Figura 7-5.

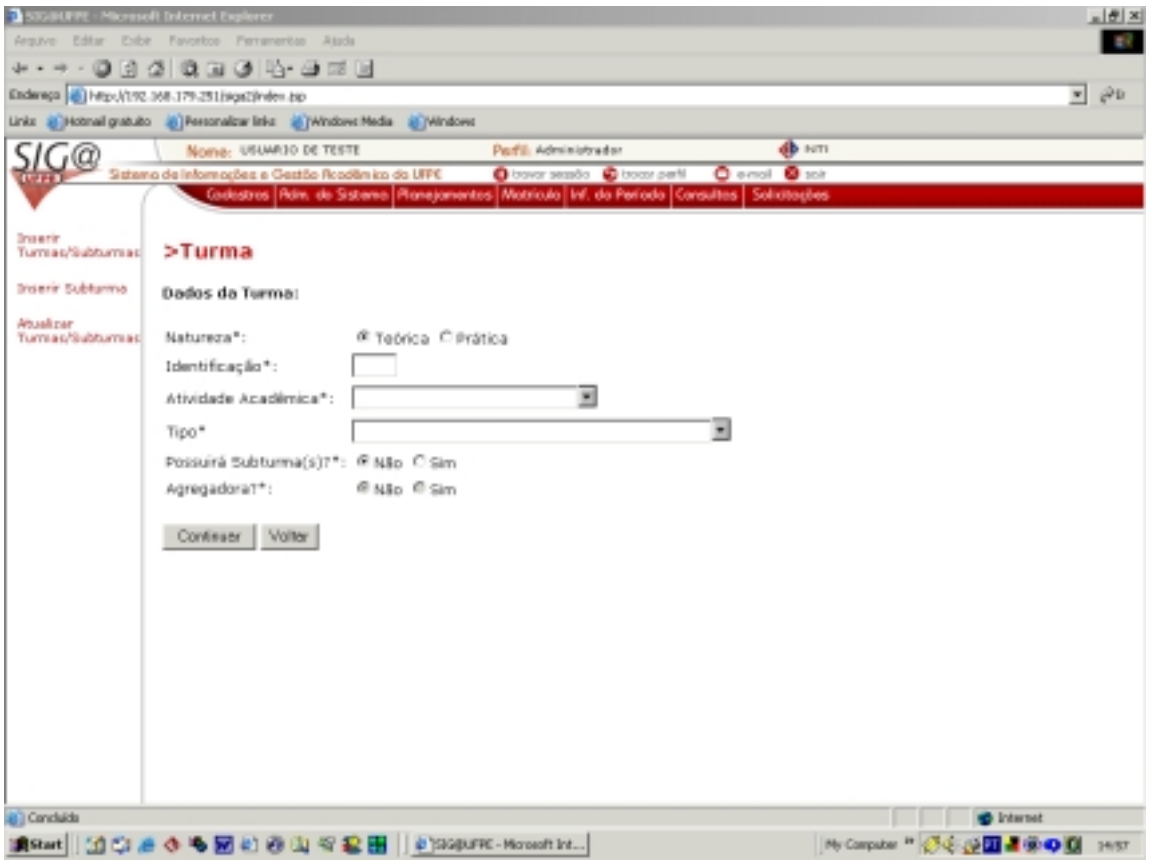


Figura 7-6 Interface Gráfica – "Especificar Turma"

A seguir, são apresentadas as interfaces gráficas da aplicação Web SIG@UFPE identificadas no caminho navegacional, descrito no Modelo Navegacional apresentado na Figura 7-5, para satisfazer a funcionalidade de detalhamento de uma turma e posterior consulta dos docentes ministrantes.

A Figura 7-7 mostra a interface gráfica, identificada no Modelo Navegacional apresentado na Figura 7-5 através da classe TelaPesquisarTurma, onde o usuário seleciona o componente curricular oferecido no período letivo, para auxiliar a busca em quais turmas este componente está sendo ofertado e confirma a ação clicando no botão pesquisar.

Figura 7-7 Interface Gráfica – "Pesquisar Turma"

A Figura 7-8 mostra a interface gráfica, identificada no Modelo Navegacional apresentado na Figura 7-5 através da classe IndiceTurma, que é retornada ao usuário com o resultado da pesquisa através de uma estrutura de índice de turmas. Nesta interface gráfica, o usuário seleciona um dos itens do índice para detalhamento e confirma a ação clicando no botão detalhar.

Figura 7-8 Interface Gráfica – "Índice Turma"

A Figura 7-9 mostra a interface gráfica, identificada no Modelo Navegacional apresentado na Figura 7-5 através da classe TelaDetalharTurma, que é retornada ao usuário com o detalhamento da turma selecionada. Nesta interface gráfica, o usuário pode continuar a navegação para saber maiores informações, tal como quais são os docentes ministrantes da turma, através do clique no link "Relação dos Docentes Ministrantes da Turma".

Figura 7-9 Interface Gráfica – "Detalhamento Turma"

A Figura 7-10 mostra a interface gráfica, identificada no Modelo Navegacional apresentado na Figura 7-5 através da classe IndiceDocenteMinistrante, que é retornada ao usuário com a coleção de docentes ministrantes da turma através de uma estrutura de índice.

Figura 7-10 Interface Gráfica – "Índice Docentes Ministrantes"

7.5 Considerações Finais

Neste capítulo foi apresentado o estudo de caso da aplicação Web SIG@UFPE, para validar de forma prática os benefícios da execução do subfluxo Projetar Camada de Apresentação, característica mais importante da extensão do fluxo de Análise e Projeto do RUP para o desenvolvimento de aplicações Web.

Os artefatos apresentados neste capítulo foram desenvolvidos através da ferramenta de modelagem de sistemas Rational Rose (versão 2001.03.00). Esta ferramenta possui um mecanismo para integrar a semântica proposta nas extensões de UML, como a inclusão dos novos estereótipos para serem usados em elementos específicos dos modelos. Além disto, esta ferramenta possibilita automatização na criação e/ou alteração de modelos, através da criação de scripts que manipulam uma API (*Application Program Interface*) fornecida, denominada REI (*Rational Extensibility Interface*).

No projeto SIG@UFPE o método proposto para criação do Modelo Navegacional apresentado na seção 6.3 está de fato sendo utilizado. Para acelerar o desenvolvimento, foi desenvolvida uma função (através de script) baseada neste método, integrada ao ambiente da ferramenta Rational Rose, que solicita como entrada o Modelo de Análise, e então cria automaticamente o Modelo Navegacional correspondente.

A realização do estudo de caso foi fundamental para verificação da efetividade da execução do subfluxo Projetar Camada de Apresentação, bem como para revisar e redefinir o método proposto para criação do Modelo Navegacional, e as atividades e artefatos deste subfluxo.

Capítulo 8

Conclusão

A demanda pelo desenvolvimento de aplicações Web está cada dia maior, devido a fatores econômicos, sociais e culturais. Entretanto, estas aplicações possuem características que as diferenciam de aplicações tradicionais, e que devem ser observadas e atendidas no processo de desenvolvimento.

Diante deste panorama, este trabalho visou atender necessidades específicas do desenvolvimento de aplicações Web. Para tanto, foi necessária uma adaptação na metodologia genérica de desenvolvimento de software RUP – *Rational Unified Process*, com foco no fluxo de Análise e Projeto.

Neste capítulo são apresentadas as considerações gerais e contribuições deste trabalho, bem como os trabalhos correlatos e propostas de trabalhos futuros.

8.1 Considerações Gerais e Contribuições

A Web pode ser definida como um universo de informações globalmente acessível através da Internet. É um espaço abstrato dentro do qual pessoas podem interagir. A Web marca o fim de uma era de frustração e incompatibilidade entre sistemas de computação, criando uma explosão de acessibilidade, com muitos impactos sociais e econômicos.

Do ponto de vista social, a Web age como uma ferramenta de disseminação e popularização da informação, sendo de fundamental importância para a construção de uma "cultura global" onde barreiras políticas, econômicas e éticas têm menor influência. A Web integra países, pessoas e culturas diferentes, sem distinção de raça, credo ou cor, formando uma comunidade on-line onde o objetivo é o máximo de interação [Araújo 01].

Enquanto agente de disseminação de cultura, a Web tem se tornado fundamental para a pesquisa como um adjunto ou substituto para a biblioteca. Mesmo em salas de aula, sua presença é de grande importância no auxílio de professores, aumentando a interação no ensino. Através da Internet, a Web também é usada para aprendizagem à distância, e iniciativas nesse sentido têm sido realizadas com sucesso [Kessler 99].

Entretanto, onde a Web mais tem se destacado é na economia, simplesmente porque os produtos e serviços anunciados ou ofertados possuem uma enorme audiência de possíveis compradores ou clientes. Além disso, devido em grande parte à globalização da economia mundial, cada vez mais as empresas estão migrando ou desenvolvendo seus sistemas corporativos para uma plataforma baseada na Web.

A influência positiva da Web nos mais diversos setores como economia, educação, e o entretenimento, entre outros, tem ocasionado uma demanda bastante significativa no desenvolvimento de aplicações baseadas na Web.

As aplicações Web possuem características específicas que as diferenciam das aplicações tradicionais, e que precisam ser tratadas no processo de desenvolvimento de uma forma disciplinada. Para tanto, fazem-se necessárias adaptações em processos de desenvolvimento de software já existentes para um melhor atendimento na construção de aplicações Web.

Visando atender as necessidades de desenvolvimento específicas para aplicações Web, este trabalho propôs uma adequação da metodologia de desenvolvimento de software RUP (*Rational Unified Process*), mais especificamente no fluxo de Análise e Projeto, tendo em vista que a etapa de análise e projeto é onde há uma maior diferença no desenvolvimento de aplicações Web com relação a aplicações tradicionais.

O RUP foi escolhido devido ao fato de ser genérico e adaptável e por reunir o melhor de várias técnicas modernas de desenvolvimento de software, bem como pela sua grande aceitação nos meios acadêmico e comercial.

O fluxo de Análise e Projeto do RUP foi adaptado (estendido) para considerar mais apropriadamente o desenvolvimento de aplicações Web (utilizando o padrão arquitetural de camadas) sem perder, contudo, as características de adequabilidade a outros tipos de aplicação e a generecidade do processo.

A extensão do fluxo de Análise e Projeto do RUP visou atender ao projeto da Camada de Apresentação de aplicações Web, já que esta é a camada onde estão as maiores diferenças com relação a aplicações tradicionais. Além disto, com base no framework de Análise e Projeto proposto por A.Araújo [Araújo 01], foram feitas recomendações para execução de atividades já existentes de modo a propiciar um melhor atendimento ao desenvolvimento de aplicações Web.

Desta forma, um novo subfluxo chamado Projetar Camada de Apresentação foi criado, juntamente com as atividades Projetar Navegação e Projetar GUI (*Graphic User Interface*). A atividade Projetar Navegação consiste basicamente na criação do Modelo Navegacional, com o propósito de identificar como o usuário navega (caminha) pela aplicação para utilizar as funcionalidades oferecidas. A atividade Projetar GUI consiste na criação de esboços de interfaces gráficas, geralmente páginas HTML, para os elementos do Modelo Navegacional (páginas cliente, formulários) que servem para interação do usuário com a aplicação.

Para orientar a criação do Modelo Navegacional, um método foi proposto com base nas extensões de UML propostas por J.Conallen [Conallen 99b] e N.Koch [Koch 01], e na metodologia para desenvolvimento de sistemas hipermídia proposta por H.Baumeister [Baumeister 99].

A validação do subfluxo proposto para o projeto da Camada de Apresentação foi realizada através de um estudo de caso sobre a aplicação Web SIG@UFPE (Sistema de Informações e Gestão Acadêmica da UFPE). O estudo de caso foi de fundamental importância para a verificação da efetividade do subfluxo Projetar Camada de Apresentação, bem como para revisar o método proposto para criação do Modelo Navegacional, e as atividades e artefatos deste subfluxo.

O estudo de caso, apresentado neste trabalho, é uma pequena parte integrante da aplicação Web SIG@UFPE que está sendo desenvolvida no Núcleo de Tecnologia da Informação desta universidade.

Para otimizar e agilizar o processo de desenvolvimento, a ferramenta de modelagem de sistemas Rational Rose precisou ser configurada para incorporar os estereótipos (inclusive com a representação gráfica) das extensões de UML utilizadas no Modelo Navegacional, bem como para automatizar a criação deste modelo. A automatização da criação do Modelo Navegacional baseou-se no método proposto para construção deste modelo, e consistiu da criação de uma nova função incorporada ao ambiente da ferramenta Rational Rose.

Esta função executa um programa escrito em uma linguagem de script que manipula uma API (*Application Program Interface*) chamada REI (*Rational Extensibility Interface*), fornecida pela ferramenta Rational Rose. Este programa solicita como entrada o Modelo Conceitual e então gera uma visão deste modelo com ênfase nos aspectos de navegação e de apresentação, ou seja, uma versão inicial do Modelo Conceitual que posteriormente precisa ser refinada pelo projetista navegacional da aplicação.

8.2 Trabalhos Relacionados

Tendo em vista que aplicações Web podem ser consideradas como um tipo de sistema hipermídia, diversos métodos para o desenvolvimento de sistemas hipermídia têm sido propostos. Em [Koch 99] é apresentado um estudo comparativo sobre vários desses métodos. Entre os métodos mais relevantes, estão:

- HDM (*Hypermedia Design Method*) – um dos primeiros métodos desenvolvidos para definir a estrutura e interação na aplicações hipermídia [Garzotto 93], é baseado na notação E-R (Entidade e Relacionamento), mas estende o conceito de entidade e introduz novas primitivas como unidades (nós) e links.

- RMM (*Relationship Management Methodology*) – o seu propósito é melhorar o projeto e a construção de aplicações hipermídia, definindo para tanto um processo de sete passos [Isakowitz 95]. Estes passos são: projeto de entidade-relacionamento, projeto de unidades de apresentação, projeto navegacional, projeto de interface do usuário, projeto de conversão, projeto de comportamento em tempo de execução, e construção e teste.

- OOHDM (*Object-Oriented Hypermedia Design Method*) – compreende as atividades de modelagem conceitual, projeto navegacional, projeto de interface abstrata e implementação [Rossi 96,Schwabe 98]. Estas atividades são realizadas em um misto de estilos de desenvolvimento incremental, iterativo e baseado em protótipo.

- WSDM (*Web Site Design Method*) – é uma abordagem centrada no usuário que define os objetos de informação com base nos requisitos de informação dos usuários de uma aplicação Web [De Troyer 97]. Consiste de três fases principais: modelagem de usuário, projeto conceitual e projeto de implementação.

- RNA (*Relationship-Navigational Analysis*) – define uma seqüência de passos a serem usados para o desenvolvimento de aplicações Web, focando na análise [Bieber 98]. É especialmente útil para aplicações Web criadas com base em sistemas legados.

- UPHD (*Unified Process-based Hypermedia Systems Development*) – metodologia para o desenvolvimento de sistemas hipermídia [Koch 00], baseada no Processo Unificado [Jacobson 99].

O nosso trabalho procurou adaptar uma metodologia genérica de desenvolvimento de softwares para o caso específico de aplicações Web. Para tanto, escolhemos o RUP e, focando no fluxo de Análise e Projeto, criamos novas atividades e artefatos. O principal artefato proposto foi o Modelo Navegacional, e para orientar sua construção utilizamos alguns conceitos do OOHDM, tais como primitivas de acesso e caminhos navegacionais.

8.3 Perspectivas e Trabalhos Futuros

Parte da extensão do fluxo de Análise e Projeto do RUP consistiu na criação de um novo subfluxo para tratar o projeto da Camada de Apresentação, juntamente com suas atividades e artefatos. Uma das atividades do novo subfluxo, denominada Projetar GUI (*Graphic User Interface*), tem como propósito a criação de esboços das interfaces gráficas para interação do usuário com a aplicação, com base nas necessidades navegacionais. Entretanto, esta atividade pode ser refinada para tratar questões relativas à organização e aparência da interface, utilizando, para tanto, técnicas específicas para interação entre homem e máquina.

Ao final da execução do subfluxo proposto para o projeto da Camada de Apresentação, temos os subsídios necessários para a criação desta camada no fluxo de Implementação. Além disso, o Modelo Navegacional que é um dos artefatos produzidos, pode ter seus elementos (página cliente, página servidor, formulário, entre outros) mapeados em elementos de um documento XML – *Extensible Markup Language* [W3C 02], e este documento pode servir de entrada para uma ferramenta ou ambiente que automatize a criação da Camada de Apresentação de aplicações Web. Uma iniciativa como esta está sendo realizada no projeto realizado no Núcleo de Tecnologia da Informação desta universidade, responsável pela construção da aplicação Web SIG@UFPE.

Este trabalho adaptou apenas o fluxo de Análise e Projeto do RUP, devido ao fato de que a etapa de análise e projeto é onde estão as maiores diferenças no desenvolvimento de aplicações Web com relação às aplicações tradicionais. Entretanto, os demais fluxos de desenvolvimento do RUP também necessitam ser adaptados para torná-los mais adequados ao desenvolvimento de aplicações Web. Além disso, seria interessante um estudo sobre as necessidades de adequação dos fluxos de suporte (gerenciamento de projeto, gerenciamento de configuração e mudanças, e ambiente).

Outros processos genéricos como o Open [Sellers 97] e o Catalysis [D'Sousza 98] que se propõem a guiar o desenvolvimento de aplicações também necessitam ser adaptados para casos mais específicos, como o desenvolvimento de aplicações Web.

Referências Bibliográficas

[Araújo 01] A. Araújo. Framework de Análise e Projeto Baseado no RUP para o Desenvolvimento de Aplicações Web, Dissertação de Mestrado, UFPE, Centro de Informática, 2001.

[Baresi 01] L. Baresi, F. Garzotto, P. Paolini. Extending UML for Modeling Web Applications, 34th Hawaii International Conference on System Sciences, USA, 2001.

[Baumeister 99] H. Baumeister, N. Koch, L. Mandel. Towards a UML Extension for Hypermedia Design, In Proceedings <<UML>>'99, France, Vol. 1723, pp. 614-629, 1999.

[Bennett 99] S. Bennett, S. McRobb, R. Farmer. Object-Oriented Systems Analysis and Design using UML. McGraw-Hill Publishing Company, 1999.

[Beveridge 98] T. Beveridge, P. McGlashan. Programação de Alto Desempenho na Web com ISAPI e NSAPI. Tradução M. Pinto. Editora Berkeley, 1998.

[Bieber 98] M. Bieber, R. Galnares, Q. Lu. Web Engineering and Flexible Hypermedia. In Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia, Hypertext, 1998.

[Booch 99] G. Booch, J. Rumbaugh, I. Jacobson. The Unified Modeling Language User Guide, Addison Wesley Object Technology Series, 1999.

[Borba 00] P. Borba, V. Alves. Desenvolvendo Aplicações Distribuídas em Java. UFPE – Centro de Informática, Relatório Técnico, Maio, 2000.

[Clemons 91] E. Clemons, M. Row. Information Technology at Rosenbluth Travel: Competitive advantage in a rapidly growing global service company, J. Manage, Info. 8, pp. 53-79, 1991.

[Conallen 99a] J. Conallen. Modeling Web Applications Architectures with UML, Communication of the ACM, Vol. 42, No. 10, pp. 63-70, October, 1999.

[Conallen 99b] J. Conallen. Building Web Applications with UML, Addison Wesley Object Technology Series, 1999.

[D'Sousza 98] D. D'Sousza, A. Wills. Objects, Components and Frameworks with UML: The Catalysis Approach, Addison Wesley, Object Technology Series, 1998.

[De Troyer 97] O. De Troyer, C. Leune. WSDM: A User-Centered Design Method for Web Sites. In Proceedings of the 7th International World Wide Web Conference, 1997.

[Fielding 97] R. Fielding, G. Kaiser. The Apache HTTP server project, IEEE Internet Computing, July, pp. 88-90, 1997.

[Garzotto 93] F. Garzotto, P. Paolini, D. Schwabe. HDM – A Model-Based Approach to Hypertext Application Design, ACM Transactions of Information Systems, pp. 1-26, 1993.

[Gibbs 94] W. Gibbis. Software's chronic crisis, Scientific American, September, 1994.

[Hanneghan 96] M. Hanneghan. The World-Wide Web as a Platform for Supporting Interactive Concurrent Engineering. Proceedings of the 8th International Conference of Advanced Information Systems Engineering, 1996.

[Hansen 99] S. Hansen, Y. Deshpande, S. Murugusan. A Skills Hierarchy for Web Information System Development, First Workshop on Web Engineering in International Conference on Software Engineering, USA, May, pp. 16-22, 1999.

[Hech 98] E. Hech, P. Vervest. How should CIOs deal with Web-based Auctions, Communications of the ACM, Vol. 41, No. 10, pp. 99-100,July, 1998.

[Hennicker 00] R. Hennicker, N. Koch. A UML – based Methodology for Hypermedia Design, In Proceedings <<UML>>'00, USA, 2000.

[Isakowitz 95] T. Isakowitz, E. Stohr, P. Balasubramanian. A Methodology for Design of Structured Hypermedia Applications. Communications of the ACM, Vol. 38, No. 8, pp. 149-160, 1995.

[Jacobson 99] I. Jacobson, G. Booch, J. Rumbaugh. The Unified Software Development Process. Addison Wesley, 1999.

[Kessler 99] G. Kessler, K. Rosenblad, S. Shepard. The Web can be suitable for Learning?, IEEE Computer, pp. 88-90, February, 1999.

[Koch 00] N. Koch. Hypermedia Systems Development based on the Unified Process, Technical Report 003, Ludwig-Maximilians-University Munich, 2000.

[Koch 01] N. Koch, H. Baumeister, R. Hennicker, L. Mandel. Extending UML to Model Navigation and Presentation in Web Applications, 2001.

[Koch 99] N. Koch. A Comparative Study of Methods for Hypermedia Development, Technical Report 9901, Ludwig-Maximilians-University Munich, 1999.

[Nambisan 99] S. Nambisan, Y. Wang. Roadblocks to Web Technology, Communications of the ACM, Vol. 42, 1999.

[Pressman 92] R. S. Pressman. Software Engineering: A Practitioner's Approach, McGraw-Hill, 3ª ed., 1992.

[Rossi 96] G. Rossi. OHDM: Object-Oriented Hypermedia Design Method, PhD Thesis, PUC-Rio, Brasil, 1996.

[Rossi 99] G. Rossi, D. Schwabe, F. Lyardet. Web Application Models are more than Conceptual Models, In Proceedings of the World Wide Web and Conceptual Modeling'99 Workshop, ER'99 Conference, Paris, 1999.

[Schwabe 96] D. Schwabe, G. Rossi, S. Barbosa. Systematic Hypermedia Design with OOHDM, In Proceedings of the ACM International Conference, 1996.

[Schwabe 98] D. Schwabe, G. Rossi. Developing Hypermedia Applications using OOHDM. In Proceedings of Workshop on Hypermedia Development Process, Methods and Models, Hypertext, 1998.

[Sellers 97] B. Sellers, H. Younessi, I. Graham. The Open Process Specification. Addison Wesley, Open Series, 1997.

[Sommerville 96] I. Sommerville. Software Engineering – 5th Edition, Addison Wesley, 1996.

[Souza 01] R. Souza. Extensões de UML para Aplicações Web, Trabalho Individual de Engenharia de Software – UFPE/CIN, 2001.

[Sun 01] Sun – http://java.sun.com, Last acess in September/2001.

[Vasudevan 96] S. Vasudevan, Y. Wang. Technology adoption in the presence knowledge barriers: The Case of the World Wide Web. In Proceedings of the 17th International Conference on Information Systems, December, 1996.

[W3C 02] World Wide Web Consortium – http://www.w3c.org, Last acess in January/2002.

[Weippl 00] E. Weippl. Web Engineering for Intranets, Software Competence Center Hagenberg (http://www.scch.at), 2000.

[Zelnick 98] N. Zelnick. Nifty Technology and Nonconformance: The Web in Crisis, IEEE Computer, pp. 115,116,119, October, 1998.

Apêndice A

Guia de Projeto Navegacional

O Projeto Navegacional visa atender os aspectos de navegação e de apresentação que são bastantes relevantes em se tratando de aplicações Web. Para tanto, o Projeto Navegacional deve produzir o Modelo Navegacional da aplicação orientado por algumas regras.

Desta forma, neste apêndice é apresentado um Guia de Projeto Navegacional que define regras para criação do artefato Modelo Navegacional.

Introdução

O Projeto Navegacional consiste na criação do Modelo Navegacional de uma aplicação Web, com os seguintes propósitos: identificar como o usuário navega na aplicação para utilizar as funcionalidades oferecidas, criar subsídios para o projeto das interfaces gráficas do usuário, e identificar os elementos necessários para criação da Camada de Apresentação da aplicação.

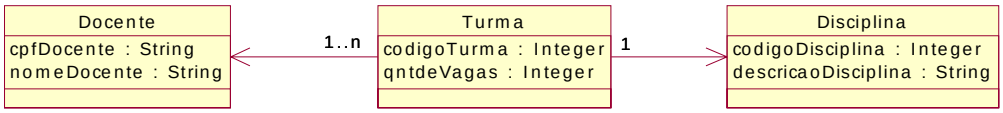
Entretanto, a criação de um Modelo Navegacional é uma atividade complexa que deve orientar-se por regras definidas em um processo. Desta forma, a seguir é apresentado o processo proposto para criação de um Modelo Navegacional com base no Modelo de Análise de uma aplicação.

Processo para criação do Modelo Navegacional

O Modelo Navegacional é uma visão do Modelo de Análise, com ênfase nos aspectos de navegação e de apresentação. Desta forma, as classes navegacionais são derivadas das classes básicas do Modelo de Análise e os caminhos navegacionais são derivados das associações entre as classes básicas do Modelo de Análise.

Os caminhos navegacionais representam como o usuário caminha (navega) na aplicação para utilizar as funcionalidades oferecidas pelo sistema. Os caminhos navegacionais são influenciados pelas multiplicidades das associações entre as classes básicas do Modelo de Análise. Além disso, os caminhos navegacionais devem satisfazer as operações de manutenção (inclusão, alteração, exclusão e consulta) de objetos do sistema.

Para exemplificar o processo de criação de um Modelo Navegacional, consideramos parte de um Modelo de Análise conforme apresentado na próxima figura. Este exemplo considera os dois tipos básicos de multiplicidade (igual a um e maior que um) em associações entre classes de um Modelo de Análise que influenciam na criação de um Modelo Navegacional.



O processo de criação do Modelo Navegacional derivado da parte do Modelo de Análise apresentado na figura acima, orienta-se por algumas regras conforme apresentado a seguir. Na próxima figura, o Modelo Navegacional é apresentado.

O Modelo Navegacional deve satisfazer as operações de manutenção (inclusão, pesquisa, alteração e exclusão) de objetos da classe Turma. Portanto, deve contemplar as seguintes considerações:

- Em termos de apresentação: criação de uma interface gráfica para inclusão de objetos da classe Turma (classe TelaIncluirTurma); criação de uma interface gráfica para auxiliar a pesquisa de objetos da classe Turma (classe TelaPesquisarTurma); criação de uma interface gráfica para exibir o resultado da pesquisa através de uma estrutura de índice (classe IndiceTurma); criação de uma interface gráfica para exibir o detalhamento, ou seja, todas as informações de um objeto da classe Turma (classe TelaDetalharTurma).

- Em termos de navegação: criação de um caminho navegacional ② para satisfazer a operação de inclusão de uma turma; criação de uma caminho navegacional ③ para satisfazer as operações de pesquisa, detalhamento, alteração e exclusão de uma turma (estas operações estão agrupadas em apenas um caminho navegacional pois para alterar e/ou excluir uma turma, tem-se que primeiro pesquisar e detalhar esta turma).

A associação entre a classes Turma e Disciplina possui multiplicidade igual a um na origem, o que significa que um objeto da classe Turma possui como atributo um objeto da classe Disciplina. Desta forma, o Modelo Navegacional deve contemplar as seguintes considerações:

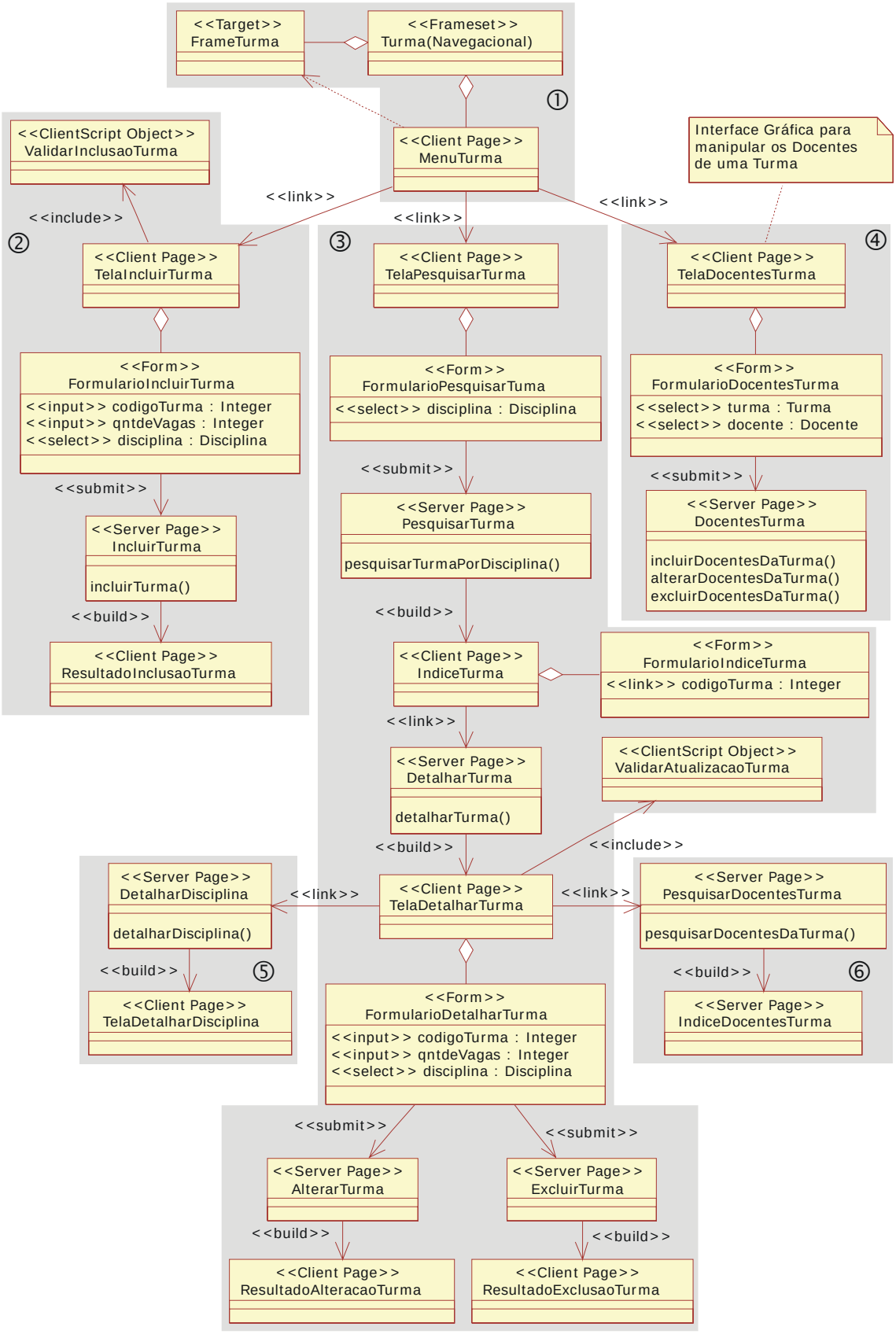
- Em termos de apresentação: a interface gráfica responsável pela operação de inclusão de objetos da classe Turma deve possuir um campo para identificar o objeto da classe Disciplina (atributo disciplina da classe FormularioIncluirTurma); a interface gráfica responsável pela operação de detalhamento dos objetos da classe Turma deve possuir um campo que identifique o objeto contido da classe Disciplina (atributo disciplina da classe FormularioDetalharTurma), com opção de detalhamento através de link (associação entre as classes TelaDetalharTurma e DetalharDisciplina).

- Em termos de navegação: criação de um caminho navegacional ⑤ para satisfazer a operação de detalhamento da disciplina de uma turma.

A associação entre as classes Turma e Docente possui multiplicidade maior que um na origem, o que significa que um objeto da classe Turma possui como atributo uma coleção de objetos da classe Docente. Desta forma, o Modelo Navegacional deve contemplar as seguintes considerações:

- Em termos de apresentação: deve ser criada uma interface gráfica para incluir, alterar e excluir a coleção de objetos da classe Docente que pertencem a um objeto da classe Turma (classe TelaDocentesTurma); a interface gráfica responsável pela operação de detalhamento de objetos da classe Turma deve possuir um campo do tipo link , que quando acionado execute um processo cujo propósito é o de exibir os docentes da turma através de uma estrutura de índice (associação entre as classes TelaDetalharTurma e PesquisarDocentesDaTurma).

- Em termos de navegação: criação de um caminho navegacional ④ para satisfazer as operações de inclusão, alteração e exclusão de docentes da turma; criação de um caminho navegacional ⑥ para satisfazer a operação de pesquisa dos docentes da turma.


<<Target>>
FrameTurma
<<Frameset>>
Turma(Navegacional)
①
<<ClientScript Object>>
ValidarInclusaoTurma
<<Client Page>>
MenuTurma
Interface Gráfica para manipular os Docentes de uma Turma
<<link>>
<<link>>
<<link>>
<<include>>
②
<<Client Page>>
TelaIncluirTurma
③
<<Client Page>>
TelaPesquisarTurma
④
<<Client Page>>
TelaDocentesTurma
<<Form>>
FormularioIncluirTurma
<<input>> codigoTurma : Integer
<<input>> qntdeVagas : Integer
<<select>> disciplina : Disciplina
<<Form>>
FormularioPesquisarTuma
<<select>> disciplina : Disciplina
<<Form>>
FormularioDocentesTurma
<<select>> turma : Turma
<<select>> docente : Docente
<<submit>>
<<submit>>
<<submit>>
<<Server Page>>
IncluirTurma
incluirTurma()
<<Server Page>>
PesquisarTurma
pesquisarTurmaPorDisciplina()
<<Server Page>>
DocentesTurma
incluirDocentesDaTurma()
alterarDocentesDaTurma()
excluirDocentesDaTurma()
<<build>>
<<build>>
<<Client Page>>
ResultadoInclusaoTurma
<<Client Page>>
IndiceTurma
<<Form>>
FormularioIndiceTurma
<<link>> codigoTurma : Integer
<<link>>
<<Server Page>>
DetalharTurma
detalharTurma()
<<ClientScript Object>>
ValidarAtualizacaoTurma
<<build>>
<<include>>
<<Server Page>>
DetalharDisciplina
detalharDisciplina()
<<link>>
<<Client Page>>
TelaDetalharTurma
<<link>>
<<Server Page>>
PesquisarDocentesTurma
pesquisarDocentesDaTurma()
<<build>>
⑤
<<build>>
⑥
<<Client Page>>
TelaDetalharDisciplina
<<Form>>
FormularioDetalharTurma
<<input>> codigoTurma : Integer
<<input>> qntdeVagas : Integer
<<select>> disciplina : Disciplina
<<Server Page>>
IndiceDocentesTurma
<<submit>>
<<submit>>
<<Server Page>>
AlterarTurma
<<Server Page>>
ExcluirTurma
<<build>>
<<build>>
<<Client Page>>
ResultadoAlteracaoTurma
<<Client Page>>
ResultadoExclusaoTurma

① As funcionalidades oferecidas pela classe Turma, tais como, cadastrar turma, pesquisar turma e cadastrar docentes da turma são exibidas para o usuário através de uma estrutura de menu com seus respectivos links.

② Caminho navegacional para a operação de inclusão de uma turma. O usuário visualiza a interface gráfica através de uma página cliente, preenche os campos com informações da turma e através de uma ação (clique em um botão) confirma a operação de cadastramento, são realizadas validações nas informações dos campos fornecidas pelo usuário através de scripts executados no cliente, posteriormente essas informações são validadas e processadas no servidor através de uma página servidor, e finalmente, a turma é cadastrada na base de dados do sistema e uma mensagem de sucesso é retornada ao usuário.

③ Caminho navegacional para as operações de pesquisa, alteração e exclusão de uma turma. O usuário visualiza a interface gráfica que auxilia a pesquisa (busca) de uma turma através de uma página cliente, seleciona a disciplina para saber em quais turmas ela está sendo oferecida e após isso confirma a operação de busca através de uma ação, é realizado um processamento no servidor, através de uma página servidor, utilizando o argumento (disciplina) fornecido pelo usuário, e então é exibido ao usuário o resultado da busca, através de uma estrutura de índice. O usuário pode selecionar um dos itens do índice para visualizar o detalhamento de uma turma, e desta forma é apresentada uma interface gráfica, através de uma página cliente, com seus campos preenchidos com todas as informações da turma. Após a pesquisa e detalhamento da turma, o usuário pode realizar as operações de alteração ou exclusão desta turma.

④ Caminho navegacional para as operações de inclusão, alteração e exclusão de docentes de uma turma. O usuário visualiza a interface gráfica através de uma página cliente, seleciona a turma e então pode incluir e/ou excluir um docente para esta turma.

⑤ Caminho navegacional para a operação de detalhamento da disciplina de uma turma. O usuário após realizar as operações de pesquisa e detalhamento de uma turma, pode então querer visualizar o detalhamento da disciplina oferecida por esta turma. Desta forma, na interface gráfica responsável por exibir o detalhamento de uma turma, haverá um campo do tipo link que quando acionado pelo usuário executa um processo no servidor, através de uma página servidor, cujo resultado é a exibição para o usuário de uma interface gráfica, através de uma página cliente, com seus campos preenchidos com todas as informações da disciplina oferecida pela turma.

⑥ Caminho navegacional para a operação de pesquisa dos docentes de uma turma. O usuário após realizar as operações de pesquisa e detalhamento de uma turma, pode então querer visualizar os docentes da turma. Desta forma, na interface gráfica responsável por exibir o detalhamento de uma turma, haverá um campo do tipo link que quando acionado pelo usuário executa um processo no servidor, através de uma página servidor, cujo resultado é a exibição para o usuário de uma interface gráfica com uma estrutura de índice, através de uma página cliente, com campos que identifiquem os docentes da turma.

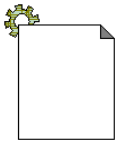
Apêndice B

A Extensão de UML WAE

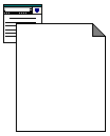
A extensão de UML WAE (*Web Application Extension*), proposta por J.Conallen [Conallen 99b], define um conjunto de estereótipos, *tagged values* e restrições que podem ser aplicados a elementos dos diagramas de UML, com o propósito de modelar características específicas de aplicações Web.

Neste apêndice, esta extensão de UML para aplicações Web é apresentada em detalhes, são descritos os estereótipos e a que elementos são aplicáveis.

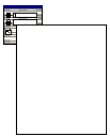
Server Page

Tipo no Metamodelo	Classe
Descrição	Uma página servidor representa uma página Web que têm scripts executados pelo servidor. Esses scripts interagem com recursos do servidor tais como, banco de dados, lógica de negócios e sistemas externos.
Ícone	
Restrições	As páginas servidor podem ter relação apenas com objetos no servidor.
Tagged Values	Engenho de script: linguagem ou engenho de precisa ser usado para executar ou interpretar esta página (*JavaScript*, *VBScript*, *Perl*, entre outros)

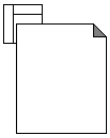
Client Page

Tipo no Metamodelo	Classe
Descrição	Uma página cliente representa uma página Web formatada, que é montada no *browser* cliente e que pode conter scripts que são interpretados pelo *browser*. Estas páginas podem ter associações com outras páginas cliente e páginas servidor.
Ícone	
Restrições	Nenhuma
Tagged Values	*TitleTag*: o título da página como exibido pelo *browser*. *BaseTag*: a URL base. *BodyTag*: o conjunto de atributos para a *tag body*.

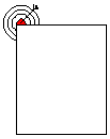
Form

Tipo no Metamodelo	Classe
Descrição	Um formulário representa uma coleção de campos de entrada que são parte da página cliente. Um formulário não têm operações, quaisquer operações que interajam com o formulário, serão propriedades da página cliente que o contêm.
Ícone	
Restrições	Nenhuma
Tagged Values	Métodos GET e POST – usados para submeter dados para ação na URL.

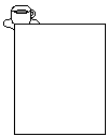
Frameset

Tipo no Metamodelo	Classe
Descrição	Container de múltiplas páginas Web. Os conteúdos de um frame podem ser uma página Web ou um outro *frameset*.
Ícone	
Restrições	Nenhuma.
Tagged Values	*Rows*: o valor do atributo linha do *tag frameset*. *Cols*: o valor do atributo coluna do *tag frameset*.

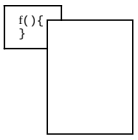
Target

Tipo no Metamodelo	Classe
Descrição	Compartimento na janela do *browser* onde páginas Web podem ser montadas. Tipicamente, um *target* é um frame na janela definido por um *frameset*. As associações *targeted link* especificam *targets* como o lugar onde uma nova página Web é montada.
Ícone	
Restrições	Um nome de *target* deve ser único para cada cliente do sistema, então, somente uma instância do *target* pode existir no mesmo cliente.
Tagged Values	Nenhum.

JavaScript Object

Tipo no Metamodelo	Classe
Descrição	No *browser* é possível simular objetos definidos pelo usuário com funções *JavaScript*. Instâncias de objetos Java script existem somente no contexto de páginas cliente.
Ícone	
Restrições	Nenhuma.
Tagged Values	Nenhum.

ClientScript Object

Tipo no Metamodelo	Classe
Descrição	Coleção separada de scripts do lado cliente, são geralmente pacotes das funções mais comuns usadas na aplicação ou na empresa.
Ícone	
Restrições	Nenhuma.
Tagged Values	Nenhum.

Link

Tipo no Metamodelo	Associação
Descrição	Um link é um ponteiro da página cliente para uma outra página Web. No diagrama de classes, um link é uma associação entre uma página cliente e outra página cliente ou uma página servidor
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	Nenhum.

Targeted Link

Tipo no Metamodelo	Associação
Descrição	Similar a associação link, um *targeted link* é um link cuja página associada é montada montada em um outro *target*.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	Parâmetros: uma lista de nomes de parâmetros que são passados junto com a requisição para a página linkada. Nome do *Target*: o nome do *target* que o link desta página aponta para ser montada nela.

Frame Content

Tipo no Metamodelo	Associação
Descrição	Agregação que expressa uma contenção do frame de outra página ou *target*. Pode apontar para outro *frameset*, indicando frames aninhados.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	*Row* e *Col*: um inteiro indicando a linha ou coluna específica do frame no *frameset* que a página associada ou *target* pertencem.

Submit

Tipo no Metamodelo	Associação
Descrição	Associação situada entre um formulário e uma página servidor. Os formulários submetem os valores de seus campos para o processamento no servidor, através de páginas servidor.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	Parâmetros: uma lista de nomes de parâmetros que devem ser passados junto com a requisição para a página linkada.

Build

Tipo no Metamodelo	Associação
Descrição	Relação que liga páginas cliente e páginas servidor. Identifica qual página servidor é responsável pela criação da página cliente.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	Nenhum.

Redirect

Tipo no Metamodelo	Associação
Descrição	Associação unidirecional com outra página Web, pode ser direcionado de e para ambas páginas cliente e páginas servidor. Se a relação origina de uma página servidor, então indica que o processamento da requisição pode continuar em outra página. Se a relação origina de uma página cliente, então indica que a página destino será automaticamente requisitada pelo *browser*.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	*Delay*: quantidade de tempo que a página cliente espera antes de redirecionar à próxima página.

IIOP

Tipo no Metamodelo	Associação
Descrição	IIOP (*Internet Inter-ORB Protocol*) é uma relação entre objetos no cliente e objetos no servidor, representa um outro mecanismo de HTTP para comunicação cliente/servidor. Tipicamente, esta relação é entre *JavaBeans* no cliente e *Enterprise JavaBeans* no servidor.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	Nenhum.

RMI

Tipo no Metamodelo	Associação
Descrição	RMI (*Remote Method Invocation*) é um mecanismo para *Java Applets* e *JavaBeans* enviarem mensagens para outros *JavaBeans* em máquinas diferentes. Tipicamente, esta relação é entre *JavaBeans* ou *Applets* no cliente e *Enterprise JavaBeans* no servidor.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	Nenhum.

Input Element

Tipo no Metamodelo	Atributo
Descrição	Atributo do objeto formulário, é usado para entrada de uma única palavra ou linha de texto.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	*Type*: o tipo de controle de entrada para ser usado – texto, número ou *password*. *Size*: especifica o tamanho da área para ser alocada na tela, em caracteres. *MaxLength*: o número máximo de caracteres que o usuário pode informar.

Select Element

Tipo no Metamodelo	Atributo
Descrição	Controle de entrada usado em formulários, permite o usuário selecionar um ou mais itens de uma lista, geralmente, expressos como *combo* ou *list box*.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	*Size*: especifica a quantidade de itens que serão exibidos ao mesmo tempo. *Multiple*: boleano que indica se mais de um item podem ser selecionados na lista.

Text Area Element

Tipo no Metamodelo	Atributo
Descrição	Controle de entrada usado em formulários, permite a entrada de múltiplas linhas.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	*Rows*: o número de linhas de textos visíveis. *Cols*: a largura do controle, de acordo com a média da largura dos caracteres.

Web Page

Tipo no Metamodelo	Componente
Descrição	Arquivos de texto acessíveis pelo servidor Web, mas podem também ser módulos compilados que são carregados e invocados pelo servidor Web. Quando acessados pelo servidor Web como um arquivo ou como um executável, uma página produz um documento HTML formatado que é enviado em resposta a requisição do *browser*.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	*Path*: o caminho requerido para especificar a página Web no servidor Web.

ASP Page

Tipo no Metamodelo	Componente
Descrição	Página Web que implementam código ASP no lado servidor. Este estereótipo é aplicável somente em ambientes de aplicações baseadas em *Active Server Pages*.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	*Path*: o caminho requerido para especificar a página Web no servidor Web.

JSP Page

Tipo no Metamodelo	Componente
Descrição	Página Web que implementam código JSP no lado servidor. Este estereótipo é aplicável somente em ambientes de desenvolvimento de aplicações Web que usam *Java Server Pages*.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	*Path*: o caminho requerido para especificar a página Web no servidor Web.

Servlet

Tipo no Metamodelo	Componente
Descrição	Componente *servlet* Java. Este estereótipo é aplicável somente em ambientes de desenvolvimento de aplicações Web que suportam *servlets*.
Ícone	Nenhum.
Restrições	Nenhuma.
Tagged Values	*Path*: o caminho requerido para especificar a página Web no servidor Web.

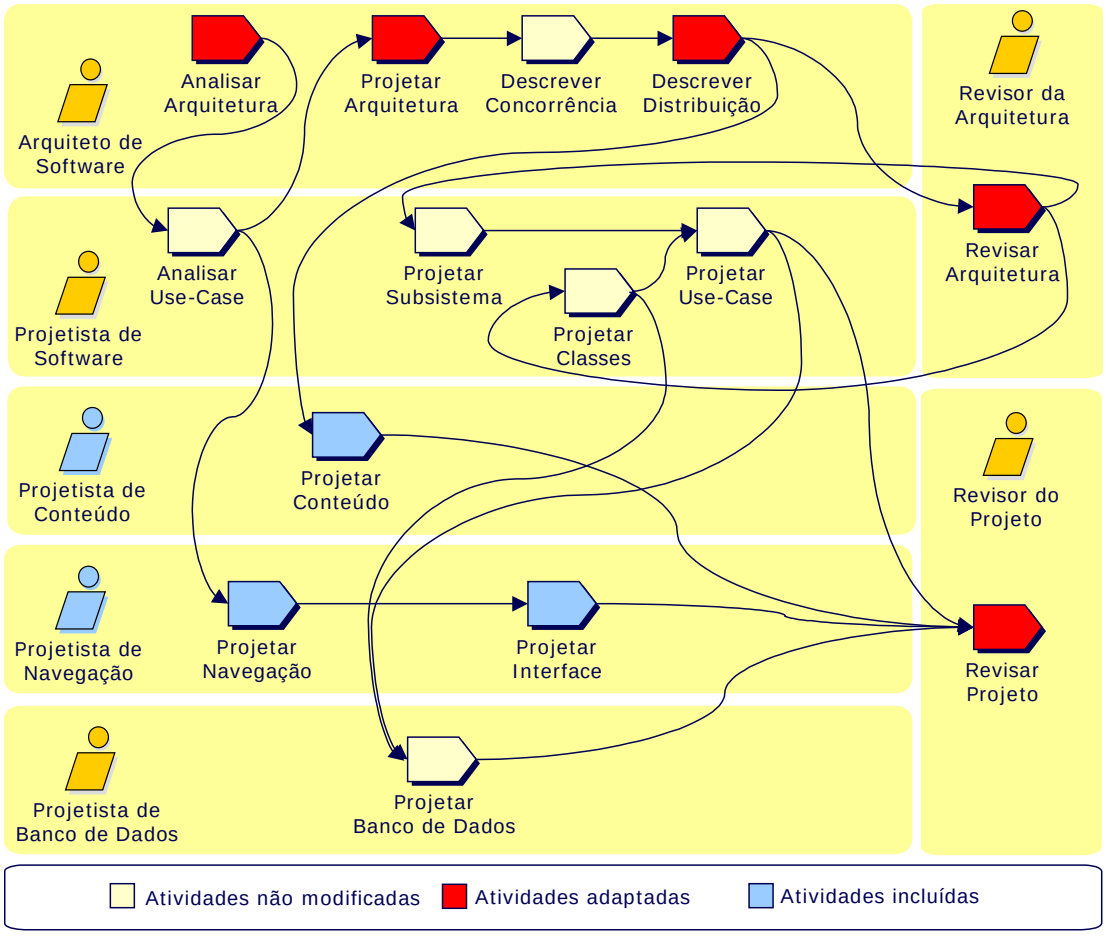
Apêndice C

Framework de Análise e Projeto para Aplicações Baseadas na Web

O framework de Análise e Projeto apresentado neste apêndice corresponde a uma adaptação do fluxo de Análise e Projeto do RUP para o desenvolvimento de aplicações Web. O fluxo foi alterado de forma a considerar as especificidades das aplicações para Web e do seu desenvolvimento. Algumas atividades do fluxo original do RUP (versão 5.0 *build* 33) não foram modificadas e, desta forma, não serão detalhadas neste apêndice.

Neste apêndice, são apresentadas as atividades, bem como seus passos correspondentes, do framework de Análise e Projeto para o desenvolvimento de aplicações baseadas na Web proposto por A.Araújo em [Araújo 01].

Framework de Análise e Projeto para aplicações Web



Os responsáveis e principais artefatos envolvidos neste fluxo são mostrados a seguir:

1. Arquiteto de Software: Modelo de Análise, Modelo de Projeto e Documento de Arquitetura de Software.
2. Projetista de Software: Modelo de Projeto (realização dos *use cases* e projeto das classes e subsistemas).
3. Revisor da Arquitetura: Modelo de Projeto e Documento de Arquitetura de Software.
4. Projetista de Conteúdo: Modelo de Conteúdo.
5. Projetista de Navegação: Modelo de Navegação e Modelo da Interface.
6. Projetista de Banco de Dados: Modelo de Dados.
7. Revisor de Projeto: Modelo de Projeto (como um todo), *Guidelines* de Projeto e *Guidelines* de Projeto para Web.

Nos parágrafos a seguir estão detalhadas apenas as atividades que foram adaptadas, as atividades incluídas e alguns *guidelines* e *checklists* que foram criados. As demais atividades, *guidelines* e *checklists*, bem como, os conceitos e mentores de ferramentas podem ser encontrados no RUP (versão 5.0 *build* 33 ano 1998).

Atividade: Analisar Arquitetura

<table>
<tr><td colspan="2">Propósito
▪ Definir os padrões arquiteturais, mecanismos chaves e convenções de modelagem para o sistema.
▪ Definir a estratégia de reuso.
▪ Providenciar informações para o processo de planejamento.
▪ Definir a estrutura inicial da aplicação baseada na Web, identificando as páginas de informação e os componentes de aplicação.</td></tr>
<tr><td colspan="2">Passos
▪ Definir Convenções de Modelagem
▪ Definir a Organização de Alto Nível dos Subsistemas
▪ Identificar os Mecanismos de Análise
▪ Identificar as Abstrações Chaves
▪ Criar Realizações de Use-Case
▪ Revisar os Resultados</td></tr>
<tr><td>Artefatos de Entrada:
▪ Modelo de Use Case
▪ Especificações Suplementares
▪ Glossário
▪ Modelo de Negócios
▪ Documento de Arquitetura de Software
▪ Modelo de Projeto
▪ Guidelines de Projeto
▪ Guidelines de Projeto para a Web</td><td>Artefatos Resultantes:
▪ Documento de Arquitetura de Software atualizado
▪ Modelo de Projeto atualizado
▪ Guidelines de Projeto
▪ Realizações dos Use Cases
▪ Guidelines de Projeto para a Web atualizadas</td></tr>
<tr><td colspan="2">Freqüência: Uma vez por iteração, durante as primeiras iterações; as iterações posteriores visitam a atividade principalmente para validar e ajustar os aspectos analíticos da arquitetura.</td></tr>
<tr><td colspan="2">Responsável: Arquiteto</td></tr>
<tr><td colspan="2">Conceitos: Padrões de Distribuição, Mecanismos de Análise, Concorrência, Divisão em Camadas</td></tr>
</table>

Para aplicações baseadas na Web os principais passos deste fluxo são "Definir a Organização de Alto Nível dos Subsistemas" e "Identificar os Conceitos Chave". Estes passos são importantes pois neles ocorrem a estruturação em camadas inicial da aplicação e a definição das classes de análise iniciais, respectivamente. Os demais passos podem não serem realizados caso o tempo seja uma questão crítica, uma constante no desenvolvimento de aplicações para a Web.

Uma atenção especial deve ser dada aos conceitos de "Definição de Camadas" e "Padrões de Distribuição" pois são de fundamental importância para a estruturação da aplicação para a Web.

Passo: Definir Convenções de Modelagem

Propósito
▪ Garantir que a representação da arquitetura e do projeto estejam consistentes através de grupos de trabalho e iterações.

Padrões e mecanismos arquiteturais, bem como o projeto de forma geral são representados de várias formas: com textos descritivos, linguagens formais, diagramas (incluindo diagramas de classe, seqüência, colaboração e atividade, entre outros), etc.

Convenções de modelagem são documentadas no Artefato: Guidelines de Projeto; o formato e o estilo das descrições arquiteturais são definidos na Seção Representação Arquitetural do Artefato: Documento de Arquitetura de Software.

O Artefato: Guidelines de Projeto para a Web apresenta algumas boas práticas para a construção de sistemas baseados na Web. Algumas convenções de modelagem e padrões arquiteturais estão previamente indicados podendo ser alterados ou adaptados à natureza específica da aplicação ou organização em questão. O Artefato: Guidelines de projeto para a Web é um documento em constante evolução devendo ser sempre atualizado para o projeto e para a organização.

Passo: Definir a Organização de Alto Nível dos Subsistemas

Propósito
▪ Criar uma estrutura inicial para o Modelo de Projeto
Guidelines: Divisão em Camadas

O modelo de projeto é normalmente organizado em camadas. O número de camadas não é fixo, mas varia de situação para situação.

Durante a análise arquitetural, o foco é normalmente as duas camadas de mais alto nível, isto é, as camadas de **aplicação** e **negócios**. As outras camadas de mais baixo nível estarão em foco durante o projeto arquitetural, ver a atividade Projetar a Arquitetura para mais informações.

Uma tendência nas aplicações atuais é a separação entre regras de negócio, apresentação (interface de usuário) e gerenciamento de dados. Esta separação de preocupações leva a uma arquitetura de aplicação de três camadas: serviços de apresentação, serviços de processamento e serviços de dados. Este padrão arquitetural é especialmente apropriado para sistemas baseados na Web pois promove a noção de interoperabilidade, reuso e gerenciamento das aplicações em ambiente distribuído. O tempo de desenvolvimento e principalmente o tempo de manutenção, de aplicações desenvolvidas a partir deste padrão, são reduzidos, adequando-se mais facilmente a característica de rápido desenvolvimento de uma aplicação Web.

Aplicações baseadas na Web podem ser simples, apenas retornando e apresentando páginas HTML ou sofisticadas aplicações corporativas que integram a Web a recursos corporativos. A definição das camadas de uma aplicação baseada na Web deve levar em conta essa diferença.

Passo: Identificar os Mecanismos de Análise

Propósito
▪ Definir os padrões arquiteturais usados pelos projetistas para dar "vida" a seus objetos.

Um mecanismo de análise representa um padrão que constitui uma solução comum para um problema comum. Eles podem ser padrões de estruturas, padrões de comportamento, ou ambos. Eles são usados durante a análise para reduzir a complexidade e para melhorar a consistência, providenciando aos projetistas pequenas representações para problemas complexos. Os mecanismos permitem que o esforço de análise seja focado sobre a tradução dos requisitos do sistema em conceitos de software, sem se preocupar com a especificação de comportamentos relativamente complexos, necessários para suportar a funcionalidade, mas não central para ela.

Mecanismos arquiteturais não são usualmente relacionados com o domínio do problema, ao contrário, são conceitos da "ciência da computação". Eles providenciam comportamento específico para classes ou componentes relacionados ao domínio, ou correspondem a implementação da cooperação entre classes e/ou componentes. Eles podem ser implementados como um *framework*. Exemplos incluem mecanismos para manusear a persistência, comunicação entre processos, manuseio de falhas ou erros, troca de mensagens, etc.

Alguns mecanismos de análise específicos para aplicações Web foram definidos e constam no artefato Guidelines de Projeto para a Web.

Passo: Identificar as Abstrações Chave

Propósito
▪ Iniciar a análise através da identificação das abstrações chave (representação de conceitos identificados durante a modelagem de negócios e a atividade de requisitos) que o sistema deve manusear.
Mentor de Ferramenta: Documentando as Classes de Análise no Rational Rose™.

As atividades de Requisitos e Modelagem de Negócios geralmente revelam os conceitos chave com os quais o sistema deve lidar; estes conceitos são manifestações das abstrações de projeto. Devido ao fato do trabalho já ter sido realizado, não há a necessidade de repetir esta identificação durante a Atividade: Analisar Use Case. Para se fazer uso do conhecimento existente, preliminarmente identifica-se as classes de entidade de análise para representar estas abstrações chaves com base no conhecimento geral do sistema, tal como os **Requisitos**, o **Glossário**, e em particular, o **Modelo do Domínio**, ou **Modelo de Negócios**, caso eles tenham sido construídos. Enquanto estamos definindo as abstrações chave, nós também definimos qualquer relacionamento que possa existir entre as classes de entidade. Devemos mostrar estas abstrações em um ou vários diagramas de classe e devemos criar uma pequena descrição para cada um.

As classes de análise identificadas neste ponto provavelmente mudarão e evoluirão durante o curso do projeto. O propósito deste passo não é identificar um conjunto de classes que sobreviverá através do projeto, mas identificar os conceitos chave que o sistema deve manusear. Não gaste muito tempo descrevendo classes de entidade em detalhes neste estágio inicial, porque há o risco de que você tenha identificado classes e relacionamentos que não são necessários para os use cases. Relembre que você encontrará mais classes de entidade e relacionamento quando estiver olhando os use cases.

Sistemas baseados na Web variam entre aplicações que apenas retornam e apresentam páginas HTML e aplicações sofisticadas que integram a Web a recursos corporativos. Devemos identificar as abstrações chave relacionadas aos componentes da aplicação e também às páginas de informação, para que as mesmas possam ser modeladas. Estas abstrações chave também podem ser obtidas a partir do domínio do problema do sistema baseado na Web que está sendo desenvolvido. As abstrações chave relacionadas às páginas de informação serão identificadas pelo mecanismo de análise "*HTML*" , conforme especificado no artefato Guidelines de Projeto para a Web. Deve-se encontrar o máximo possível de abstrações chave que representem as páginas de informação, pois estas abstrações representarão as classes de análise na próxima atividade.

Passo: Criar Realizações de Use-Case

Propósito ▪ Criar os artefatos do Modelo de Projeto usados para expressar o comportamento dos use cases.
Guidelines: Realização dos Use Cases
Mentor de Ferramenta: Usando o Rational Rose™ para criar Realizações de Use Case

Use Cases formam o foco central do trabalho inicial de análise e projeto. Para possibilitar a transição entre as atividades centradas nos Requisitos para atividades centradas no Projeto, o Artefato: Realização de Use Case serve como uma ponte, providenciando uma forma de rastrear o comportamento no Modelo de Projeto para o Modelo de Use Case, tão bem como para organizar as colaborações no Modelo de Projeto em torno do conceito de Use Case.

Para cada Use Case no Modelo de Use Case, cria-se uma Realização de Use Case no Modelo de Projeto. O nome da realização deveria ser o mesmo do Use Case associado e um traço de dependência deveria ser estabelecido da realização de use case para o use case associado.

Passo: Revisar os Resultados

Propósito ▪ Garantir que os resultados da análise arquitetural estão completos e consistentes.
Check-points: Analisar Arquitetura

Com a Análise Arquitetural concluída, revise os mecanismos arquiteturais, os subsistemas, os pacotes e a classes que foram identificadas, para garantir que eles estão completos e consistentes. Como os resultados da Análise Arquitetural são preliminares e relativamente informais, a revisão também deve ser informal.

O check-point: Analisar Arquitetura contém alguns itens de checagem específicos para sistemas baseados na Web.

Atividade Analisar Use-Case

<table>
<tr><td colspan="2">Propósito
▪ Identificar as classes que realizam o fluxo de eventos de um use case.
▪ Distribuir o comportamento do use case através destas classes, usando realizações de use case.
▪ Identificar as responsabilidades, atributos e associações das classes.
▪ Anotar o uso de mecanismos arquiteturais.</td></tr>
<tr><td colspan="2">Passos
▪ Suplementar as Descrições do Use Case
▪ Para cada realização de use case
  ▪ Encontrar Classes através do Comportamento do Use Case
  ▪ Distribuir o Comportamento do Use Case para as Classes
▪ Para cada classe de análise resultante
  ▪ Descrever Responsabilidades
  ▪ Descrever Atributos e Associações
  ▪ Qualificar os Mecanismos de Análise
▪ Unificar as Classes de Análise
▪ Avaliar seus Resultados</td></tr>
<tr><td>Artefatos de Entrada:
▪ Glossário
▪ Especificações Suplementares
▪ Guidelines para Modelagem dos Use Cases
▪ Modelo de Use Case
▪ Realizações de Use Case
▪ Documento de Arquitetura de Software</td><td>Artefatos Resultantes:
▪ Classes de Análise
▪ Modelo de Análise (opcional) ou Modelo de Projeto
▪ Realizações de Use Case atualizadas</td></tr>
<tr><td colspan="2">Freqüência: Uma vez por iteração, para um conjunto do Artefato: Realizações de Use Case</td></tr>
<tr><td colspan="2">Responsável: Projetista</td></tr>
<tr><td colspan="2">Guidelines: Workshop de Análise de Use Case, Classes de Análise.</td></tr>
<tr><td colspan="2">Mentor de Ferramenta: Usando o Rational Rose™ para capturar o resultado da Análise de Use Case</td></tr>
</table>

Em sistemas baseados na Web boa parte das classes de análise relacionadas às páginas de informação já foram identificadas durante a atividade "Análise Arquitetural" como abstrações chave e identificadas com o mecanismo de análise "HTML". Apesar disso podemos encontrar ainda algumas classes de análise a partir da análise detalhada dos use cases.

Para as classes de análise relacionadas a páginas de informação o esforço deve ser concentrado na definição dos atributos e associações.

Não é necessário executar o passo "Qualificar mecanismos de análise" para o mecanismo "HTML" definido durante a Análise Arquitetural. Este mecanismo serve unicamente para distinguir as classes referentes às páginas de informação.

Os passos, bem como o detalhamento desta atividade podem ser encontrados na versão 5.0 *build* 33 (1998) do *Rational Unified Process*.

Atividade: Projetar Arquitetura

<table>
<tr><td colspan="2">Propósito
▪ Analisar as interações das classes de análise para encontrar interfaces, classes de projeto e subsistemas de projeto
▪ Refinar a arquitetura, incorporando reuso quando possível.
▪ Identificar soluções comuns para problemas de projeto normalmente encontrados
▪ Incluir os elementos do modelo de projeto arquiteturalmente significantes na Seção de Visão Lógica do Documento de Arquitetura de Software.</td></tr>
<tr><td colspan="2">Passos
▪ Identificar Mecanismos de Projeto
  ▪ Categorizar os Clientes dos Mecanismos de Projeto
  ▪ Inventariar os Mecanismos de Implementação
  ▪ Mapear os Mecanismos de Projeto para Mecanismos de Implementação
  ▪ Documentar os Mecanismos Arquiteturais
▪ Identificar Classes e Subsistemas de Projeto
▪ Identificar Interfaces
▪ Identificar Oportunidades de Reuso
▪ Engenharia Reversa de Componentes e Banco de Dados
▪ Definir a Organização de Baixo Nível dos Subsistemas
▪ Incluir Elementos de Modelo Arquiteturalmente Significantes na Visão Lógica
▪ Check-points: Modelo de Projeto</td></tr>
<tr><td>Artefatos de Entrada:
▪ Especificações Suplementares
▪ Documento de Arquitetura de Software
▪ Modelo de Projeto
▪ Classes de Análise
▪ Guidelines de Projeto
▪ Guidelines de Projeto para a Web</td><td>Artefatos Resultantes:
▪ Modelo de Projeto (Classes, Pacotes e Subsistemas)
▪ Documento de Arquitetura de Software atualizado
▪ Guidelines de Projeto atualizadas
▪ Guidelines de Projeto para a Web atualizadas</td></tr>
<tr><td colspan="2">Freqüência: Uma vez por iteração</td></tr>
<tr><td colspan="2">Responsável: Arquiteto</td></tr>
</table>

Para sistemas baseados na Web o reuso é fundamental. Aplicações para a Web devem ser desenvolvidas rapidamente para serem competitivas. A *overengineering* deve ser evitada, usando-se componentes já existentes sempre que possível. Em particular, as bibliotecas de classes e frameworks de objetos normalmente avaliados para software OO deveriam ser utilizados sempre que possível para incrementar o reuso. Estudos comprovam que muitas vezes é melhor (mais barato e mais rápido) comprar um componente do que desenvolve-lo. Existe ainda a opção da utilização de componentes freeware. O Conceito: Reuso de Componentes traz alguns princípios para o reuso sistemático de software.

Quando do desenvolvimento de sistemas baseados na Web compostos somente de páginas de informação esta atividade não necessita ser realizada, pois não haverá a existência de subsistemas e interfaces.

Passo: Identificar Mecanismos de Projeto

Propósito ▪ Refinar os mecanismos de análise para mecanismos de projeto, baseando-se nas restrições impostas pelo ambiente de implementação.
Sub-passos: ▪ Categorizar os Clientes dos Mecanismos de Análise ▪ Inventariar os Mecanismos de Implementação ▪ Mapear os Mecanismo de Projeto para Mecanismos de Implementação ▪ Documentar os Mecanismos de Projeto
Conceitos: Mecanismos de Análise, Mecanismos de Projeto
Guidelines: Mecanismos de Projeto

Aplicações baseadas na Web estão sendo desenvolvidas usando uma combinação de tecnologias de orientação a objeto, cliente/servidor e Internet. Surge então um grande número de opções de infra-estrutura (Web *servers*, Web *gateways*, desenvolvimento baseado em Java versus CGI, *middleware* de objetos distribuídos, tecnologias de acesso a dados, configurações de rede, dados multimídia, etc.) que levam a um ilimitado número de opções tecnológicas. A implementação de uma aplicação para a Web envolve este grande número de tecnologias levando a um ambiente de implementação dos mais diversos e cheio de restrições (restrições de linguagens de programação, de mídias, de *middlewares*, de *browsers*, plataforma do servidor, etc.). Estas restrições devem ser bem definidas antes que os mecanismos de análise sejam mapeados para mecanismos de projeto e estes para mecanismos de implementação. O mecanismo de análise "HTML" não necessita ser mapeado para nenhum mecanismo de projeto e conseqüentemente para nenhum mecanismo de implementação.

Sub-Passo: Categorizar os Clientes dos Mecanismos de Projeto

Identifique os clientes de cada mecanismo de análise. Analise todos os clientes de um dado mecanismo de análise, procurando por características do mecanismo que eles necessitam.

Identifique os perfis de características de cada mecanismos de análise. Pode haver uma grande variedade de perfis de características, fornecendo vários graus de desempenho, segurança, economia de custo, etc.

Agrupe os clientes de acordo com seu uso dos perfis de características. Forme grupos de clientes que parecem compartilhar uma necessidade, de um mecanismo de análise, com características similares; identifique um mecanismo de projeto baseado em tal necessidade.

Sub-Passo: Inventariar os Mecanismos de Implementação

Faça um inventário dos mecanismos de implementação que você tem a sua disposição:

- Mecanismos oferecidos por produtos de *middleware*.
- Mecanismos oferecidos por sistemas operacionais.
- Mecanismos oferecidos por um componente.
- Mecanismos oferecidos por uma biblioteca de classes.
- Código legado (ver Engenharia Reversa de Componentes e Banco de Dados abaixo).
- Pacotes de propósito especial: Geradores de GUI, Sistemas de Informação Geográfica, SGBD.

Determine onde os mecanismos existentes podem ser usados e onde novos mecanismos de implementação necessitam ser construídos.

Sub-Passo: Mapear os Mecanismos de Projeto para Mecanismos de Implementação

Mecanismos de projeto providenciam uma abstração dos mecanismos de implementação, ligando a brecha que existe entre os mecanismos de análise e os mecanismos de implementação. O uso de mecanismos arquiteturais abstratos durante o projeto permite-nos um forma de se utilizar mecanismos arquiteturais sem obscurecer o problema com detalhes de um mecanismo em particular. Ele também permite-nos substituir potencialmente um mecanismo de implementação específico por outro sem afetar adversamente o projeto.

Determine as variações das características. Pegue as características identificadas para os mecanismos de projeto e determine as variações de seus valores.

Considere o custo de aquisição de componentes. Para mecanismos de implementação candidatos, considere o custo de aquisição ou licenciamento, a maturidade do produto, o relacionamento com o seu vendedor, o suporte, etc., além dos critérios técnicos.

Conduza a procura por componentes corretos, ou construa os componentes. Você freqüentemente perceberá que não há mecanismos de implementação apropriados para alguns mecanismos de projeto; isto disparará uma procura por um certo produto, ou identificará a necessidade do seu desenvolvimento.

A escolha do mecanismo é baseada não somente sobre uma boa solução para as características técnicas, mas também sobre características não técnicas, tais como custo. Algumas das escolhas pode ser provisórias; quase sempre todas terão algum risco associado: desempenho, robustez, escalabilidade, são sempre preocupações que devem ser validadas através de avaliação, prototipação exploratória ou inclusão em um protótipo arquitetural.

Sub-Passo: Documentar os Mecanismos Arquiteturais

Os mecanismos, o mapeamento entre eles e os detalhes sobre seu uso, devem ser documentados no Artefato: Guidelines de Projeto específico para o projeto.

Passo: Identificar Classes e Subsistemas de Projeto

Propósito ▪ Refinar as classes de análise, categorizando-as em classes de projeto e subsistemas.
Guidelines: Subsistemas de Projeto, Pacotes de Projeto, Mapeando Classes de Análise para Classes de Projeto, Representando Interfaces para Sistemas Externos, Representando Interfaces Gráficas de Usuário.
Mentores de Ferramentas: Usando o Rational Rose para Gerenciar Classes, Usando o Rational Rose para Gerenciar Subsistemas, Usando o Rational Rose para Gerenciar o Modelo de Projeto.

A Atividade: Analisar Use Case resulta em **classes de análise**, que representam conceitos que realizam um comportamento. No projeto, **classes de análise** evoluem para classes de projeto e subsistemas, refletindo a maior profundidade do projeto tão bem como a imposição de restrições impostas pelo ambiente de implementação. Um subsistema é, efetivamente, um tipo especial de pacote que tem somente interfaces como elementos públicos. As interfaces providenciam uma camada de encapsulamento, permitindo que o projeto interno do subsistema possa permanecer escondido dos outros elementos do modelo.

Identifique as Classes. Quando a classe de análise e simples e representa uma única abstração lógica, ela pode ser mapeada diretamente para uma classe de projeto. Tipicamente, classes de entidade sobrevivem relativamente intactas para o projeto. Desde que classes de entidade são tipicamente persistentes, determine se as classes de projeto devem ser persistentes

Quando estiver identificando classes, elas devem ser agrupadas no Artefato: Pacotes de Projeto, para propósitos de organização e de gerenciamento de configuração. Ver as Guidelines: Pacotes de Projeto para mais informação sobre como tomar decisões de empacotamento.

Identifique os Subsistemas. Quando a classe de análise for complexa, ela deve ser mapeada para um subsistema de projeto. Um subsistema é logicamente equivalente ao Artefato: Componente no Artefato: Modelo de Implementação.

A decisão de criar um subsistema a partir de um conjunto de colaborações de classes de análise é baseada na experiência; a representação atual pode levar umas poucas iterações para se estabilizar.

Passo: Identificar Interfaces

Propósito
▪ Identificar as interfaces dos subsistemas baseando-se em suas responsabilidades.
Mentor de Ferramenta: Usando o Rational Rose™ para representar Interfaces

Identifique um conjunto de interfaces candidatas para todos os subsistemas. Para cada subsistema, considere suas responsabilidades: organize-as em grupos de responsabilidades relacionadas e coesivas. Estes grupos definem as interfaces iniciais para o subsistema. De início, identifique uma operação para cada responsabilidade.

Procure por similaridades entre as interfaces. De um conjunto candidato de interfaces, procure por nomes similares, responsabilidades similares e operações similares. Onde as mesmas operações existirem em várias interfaces, reconstrua as interfaces, extraindo as operações comuns para uma nova interface. Esteja certo que você esta utilizando o reuso quando possível.

Defina as operações. Para as novas interfaces, defina os nomes das operações e os parâmetros baseados nas responsabilidades do subsistema.

Defina as dependências da interface. Os parâmetros de cada operação da interface tem um tipo particular: eles podem realizar um interface particular ou eles podem ser instâncias de um tipo de dado simples. Nos casos onde os parâmetros são objetos que realizam um interface em particular, defina associações de dependência entre a interface e as interfaces das quais ela depende.

Mapeie as interfaces para os subsistemas. Uma vez que as interfaces foram identificadas, crie associações de realização entre os subsistemas e as interfaces que eles realizam.

Empacote as interfaces. Se cada interface é realizada por um único subsistema, a interface pode ser colocada dentro do subsistema. Se a interface é realizada por mais de um subsistema, ela deve ser colocada dentro de um pacote em separado. Isto permite que as interfaces sejam gerenciadas e controladas independentemente dos subsistemas.

Passo: Identificar Oportunidades de Reuso

Propósito
▪ Identificar onde os subsistemas e/ou componentes pode ser reusados, baseando-se em suas interfaces.
Guidelines: Interfaces

O Conceito: Reuso de Componentes traz alguns princípios para o reuso sistemático de software.

Procure por subsistemas ou componentes existentes que ofereçam interfaces similares. Compare cada interface identificada com as interfaces providenciadas por subsistemas ou componentes existentes.

Modifique as interfaces recentemente identificadas para aperfeiçoar o encaixe. Pode haver oportunidades para que mudanças secundárias nas interfaces candidatas aumentem sua conformidade com as interfaces existentes.

Substitua interfaces candidatas por interfaces existentes quando possível. Depois da simplificação e da reconstrução, se houver igualdade com alguma interface existente, elimine a interface candidata e simplesmente use a interface existente.

Mapeie o subsistema candidato para componentes existentes. Quando um subsistema candidato pode ser realizado por um componente existente, crie rastreamento entre o subsistema de projeto e o componente no modelo de implementação.

No mapeamento de subsistemas para componentes, considere os mecanismos de projeto associados ao subsistema; requisitos de desempenho ou segurança podem desqualificar um componente.

Passo: Engenharia Reversa de Componentes e Banco de Dados

Propósito
▪ Incorporar elementos de modelo, potencialmente reusáveis, de outros projetos, fontes externas ou iterações anteriores.
Guidelines: Engenharia Reversa de Banco de Dados Relacional, Mapeando um Modelo de Objetos para um Banco de Dados Relacional.
Mentor de Ferramenta: Usando o Rational Rose™ para realizar Engenharia Reversa de Código

Baseando-se sobre os resultados do passo Identificar Oportunidades de Reuso, código e definições de bancos de dados existentes podem ser utilizados. Usando-se as oportunidades de reuso como um filtro, o trabalho de realizar a engenharia de forma reversa pode ser focado sobre os componentes que foram reusados na iteração atual.

Engenharia Reversa de Componentes

Em organizações que constroem sistemas similares, há freqüentemente um conjunto de componentes comuns que providenciam muitos dos mecanismos arquiteturais necessários para um novo sistema. Pode haver também componentes avaliados no mercado que atendem as necessidades. Componentes existentes deveriam ser examinados para determinar sua conveniência e compatibilidade dentro da arquitetura do software.

Componentes existentes, ou desenvolvidos durante interações anteriores, mas ainda não incluídos no Modelo de Projeto, ou componentes comprados, devem passar pelo processo de engenharia reversa e serem incorporados no modelo de projeto. No modelo de projeto, cada componente é representado como um subsistema com uma ou mais interfaces.

Engenharia Reversa de Banco de Dados

Bancos de dados e os dados residentes neles, representam uma das mais importantes fontes de recursos reusáveis. Para reusar as definições de classes incluídas nos bancos de dados existentes, determine que informação usada pela aplicação já existe em algum banco de dados. Realize a engenharia reversa de um conjunto de classes para representar as estruturas do banco de dados que manuseia esta informação (para mais informação, veja a Guideline: Engenharia Reversa de Banco de Dados Relacional). Construa um mapeamento entre a representação da classe na aplicação e a estrutura usada no banco de dados. Para o mapeamento entre classes e tabelas em um banco de dados relacional, veja a Guideline: Mapeando um Modelo de Objetos para um Banco de Dados Relacional.

Passo: Definir a Organização de Baixo Nível dos Subsistemas

Propósito
▪ Organizar as camadas inferiores no Modelo de Projeto.
Guidelines: Divisão em Camadas, Subsistemas de Projeto, Pacotes de Projeto
Mentor de Ferramenta: Usando o Rational Rose para Gerenciar o Modelo de Projeto

As camadas inferiores da arquitetura providenciam a interface para a infra-estrutura do sistema. Os pacotes e subsistemas de mais baixo nível providenciam uma forma para estruturar o modelo de projeto e para aperfeiçoar sua estabilidade, legibilidade e flexibilidade. A divisão em camadas pode, também, providenciar um forma para reduzir o impacto de mudanças: através de regras que restringem as dependências entre pacotes e subsistemas, reduzindo o grau de acoplamento e tornando, desta forma, o sistema mais robusto.

Atribua as responsabilidades de subsistemas e camadas para indivíduos ou equipes. Cada pacote ou subsistema deveria ser de responsabilidade de uma única pessoa ou equipe.

Passo: Incluir Elementos de Modelo Arquiteturalmente Significantes na Visão Lógica

Propósito ▪ Documentar os resultados do Projeto Arquitetural
Guideline: Visão Lógica
Mentor de Ferramenta: Usando o Rational Rose™ para Gerenciar o Modelo de Projeto

Quando classes, pacotes e subsistemas (elementos de modelo) são importantes em uma perspectiva arquitetural, eles deveriam ser incluídos na Seção Visão Lógica do Documento de Arquitetura de Software. Isto tornará o modelo de projeto homogêneo e ao mesmo tempo tornará a arquitetura consistente.

Atividade: Descrever a Concorrência

Propósito ▪ Definir requisitos de concorrência, identificar processos, identificar mecanismos de comunicação entre processos, alocar recursos de coordenação entre processos, identificar ciclo de vida de processos e distribuir elementos de modelo entre os processos.	
Passos ▪ Definir os Requisitos de Concorrência ▪ Identificar Processos ▪ Identificar o Ciclo de Vida dos Processos ▪ Identificar Mecanismos de Comunicação entre Processos ▪ Alocar Recursos de Comunicação entre Processos ▪ Mapear Processos para o Ambiente de Implementação ▪ Distribuir Elementos de Modelo entre os Processos	
Artefatos de Entrada: ▪ Especificações Suplementares	**Artefatos Resultantes:** ▪ Visão de Processo do Documento de Arquitetura de Software
Responsável: Arquiteto	
Guidelines: Conceito: Concorrência, Checkpoint: Visão de Processo.	
Mentor de Ferramenta: Usando o Rational Rose™ para Documentar o Modelo de Processo	

Quando do desenvolvimento de sistemas baseados na Web compostos somente de páginas de informação, esta atividade não necessita ser realizada. Não haverá a necessidade de definição de concorrência ou identificação de processos.

Os passos, bem como o detalhamento desta atividade podem ser encontrado na versão 5.0 build 33 (1998) do *Rational Unified Process*.

Atividade: Descrever a Distribuição

<table>
<tr><td colspan="2">Propósito
▪ Descrever como a funcionalidade do sistema será distribuída através dos nós físicos. Necessária somente para sistemas distribuídos.</td></tr>
<tr><td colspan="2">Passos
▪ Definir a Configuração da Rede
  ▪ Levantar a Estrutura de Comunicação
  ▪ Analisar Requisitos e Características que impactam a Arquitetura do Sistema
  ▪ Definir Arquitetura do Sistema
▪ Alocar os Processos para os Nós
▪ Avaliar seus Resultados</td></tr>
<tr><td>Artefatos de Entrada:
▪ Visão de Processo do Documento de Arquitetura de Software
▪ Modelo de Implementação
▪ Guidelines de Projeto para a Web</td><td>Artefatos Resultantes:
▪ Visão de Implantação do Documento de Arquitetura de Software</td></tr>
<tr><td colspan="2">Responsável: Arquiteto</td></tr>
<tr><td colspan="2">Conceitos: Padrões de Distribuição</td></tr>
<tr><td colspan="2">Mentor de Ferramenta: Usando o Rational Rose™ para Documentar a Visão de Implantação</td></tr>
</table>

Para sistemas baseados na Web esta atividade também tem o objetivo de localizar o sistema dentro da Internet, ou seja, determinar o site, o backbone, além de determinar questões como espelhamento, segurança a nível de rede (firewalls e servidores proxy), etc.

A distribuição de processos através de dois ou mais nós requer uma examinação detalhada dos padrões de comunicação entre processos do sistema. Freqüentemente, há uma visão de que a distribuição de processos pode deslocar a carga de trabalho de uma máquina para outra. Na prática, a carga de trabalho da comunicação entre processos adicional pode facilmente negar qualquer ganho de distribuição, caso o processo e as fronteiras dos nós não sejam considerados cuidadosamente.

Mesmo assim, há muitos casos onde a carga de trabalho do sistema não pode ser manuseada por um único processador. Isto pode acontecer devido aos requisitos de distribuição. Pode também ser resultado de preocupações com a escalabilidade, onde um grande número de usuários concorrentes não pode ser suportado por um único processador ou pode resultar de preocupações econômicas, onde o preço do desempenho é menor.

A descrição da distribuição em sistemas baseados na Web é uma atividade crítica. A Internet é uma rede bastante complexa e qualquer sistema que funcione sobre a mesma deve considerar uma série de questões para que possa ter o desempenho, segurança, confiabilidade e disponibilidade necessárias. Devemos considerar questões como: desempenho do backbone, balanceamento de carga, segurança a nível de rede (firewall e proxy), espelhamento e replicação, redundância de dados e de processamento, interoperabilidade e interface com outros sistemas, etc. Estas preocupações são fundamentais na definição da infra-estrutura ou arquitetura de sistema para uma aplicação baseada na Web.

Passo: Definir a Configuração da Rede

Propósito
▪ Entender a configuração e a topologia da rede.
Sub-passos:
▪ Levantar a Estrutura de Comunicação
▪ Analisar Requisitos e Características que impactam a Arquitetura do Sistema
▪ Determinar Arquitetura do Sistema

Os sub-passos deste passo referem-se a sistemas baseados na Web.

A topologia da rede e as capacidades e características dos processadores e mecanismos da rede determinarão a natureza e o grau de distribuição possível para o sistema.

As seguintes informações necessitam ser capturadas:

- Layout físico da rede, incluindo as localizações.
- Os nós da rede, suas configurações e capacidades. A configuração inclui o hardware e o software instalados nos nós, o número de processadores, a quantidade de espaço em disco, a quantidade de memória, a quantidade de swap, etc. O hardware instalado no nó pode ser representado através de "*devices*".
- A largura de banda de cada segmento da rede.
- A existência de qualquer caminho redundante na rede.
- O propósito primário do nó. Isto inclui:
  - estações de trabalho usadas pelos usuários;
  - nós servidores nos quais o processamento ocorre;
  - configurações especiais usadas para desenvolvimento e teste;
  - outros processadores especializados.

Sub-Passo: Levantar a Estrutura de Comunicação

Mapear a estrutura de comunicação da Internet, levantando os principais backbones, gateways, sub-redes, linhas de comunicação e suas principais características (taxa de transmissão de dados, disponibilidade, etc.) e as áreas alcançadas pelos mesmos.

Sub-Passo: Analisar Requisitos e Características que impactam a Arquitetura do Sistema

Durante a etapa de elicitação de requisitos alguns tipos de requisitos especiais para sistemas baseados na Web foram definidos e devem ser analisados. Neste momento devemos analisar os requisitos relacionados à comunidade de usuários à qual se destina a aplicação em desenvolvimento (localização do núcleo base de usuários, requisitos de tempo de resposta esperado, requisitos de escala e crescimento, restrições de disponibilidade, etc.) e juntamente à analise das características da aplicação determinadas durante o seu projeto, devemos eleger a sub-rede ou site mais apropriado para a implantação do sistema.

As seguintes características necessitam ser analisadas:

- taxa de transmissão de dados necessária para que a aplicação possa ter um desempenho aceitável;
- necessidade de integração com outros sistemas baseados na Web (a localização destes sistemas é muito importante);

A partir da determinação do site mais apropriado para a implantação do sistema baseado na Web em questão devemos nos preocupar com a definição de toda a arquitetura de sistema ou infra-estrutura necessária para a execução satisfatória da aplicação.

Sub-Passo: Definir a Arquitetura do Sistema

Uma vez determinado o site mais apropriado para a implantação da aplicação devemos definir toda a infra-estrutura ou arquitetura de sistema necessária para a execução satisfatória do sistema.

A topologia do sistema e as capacidades e características dos processadores e mecanismos envolvidos (clientes, servidores de aplicação e dados e outros dispositivos como firewall, servidor proxy, etc.) determinarão a natureza e grau de distribuição possível no sistema.

As seguintes informações necessitam ser definidas:

- Os nós de processamento, suas configurações e capacidades. A configuração inclui o hardware e software instalado nos nós, o número de processadores, o quantidade de espaço em disco, a quantidade de memória, a quantidade de swap, etc. O hardware instalados nos nós pode ser representados através de "*devices*".
- A taxa de transmissão entre nós.
- A existência de espelhamento/replicação (esta é uma forma de providenciar capacidades de tolerância a falhas).
- O propósito primário de um nó. Isto inclui:
  - estações de trabalho usadas pelos usuários finais;
  - servidores de aplicação e servidores de dados;
  - configurações especiais usadas para desenvolvimento e teste;
  - outros processadores especializados, como firewall (filtram o tráfico não permitido) e servidores proxy (reescrevem pacotes escondendo a fonte original).

A descrição dos nós que representam as máquinas dos usuários finais deve se ater às características mínimas exigidas para a execução do sistema, características de hardware, mas principalmente de software, de forma a suportar as diversas tecnologias empregadas na implementação da aplicação.

Passo: Alocar os Processos para os Nós

Propósito
▪ Distribuir a carga de trabalho do sistema.

Processos deveriam ser alocados para nós de forma a minimizar a quantidade de tráfego na rede; processos que possuem um alto grau de interação deveriam ser colocados no mesmo nó; processos que interagem com menor freqüência podem residir em nós diferentes.

A alocação deve levar em conta:

- A capacidade do nó (em termos de memória e poder de processamento)
- Largura média de banda passante para comunicação (barramento, LANs, WANs)
- Disponibilidade de hardware e links de comunicação
- Requisitos de redundância e tolerância a falhas
- Requisitos de tempo de resposta
- Requisitos de *throughput*
- …e assim por diante.

Algumas *guidelines* para a alocação de processos para os nós, referentes a sistemas baseados na Web, podem ser encontradas no Artefato: Guidelines de Projeto para a Web.

Passo: Avaliar seus Resultados

Propósito
▪ Avaliar a visão de implantação.

Ver os Checkpoints para a Visão de Implantação.

Atividade: Revisar a Arquitetura

<table>
<tr><td colspan="2">Propósito
▪ Descobrir alguns riscos não percebidos ou desconhecidos no planejamento do sistema.
▪ Detectar alguma falha de projeto arquitetural. Falhas arquiteturais são difíceis de serem resolvidas, a maioria demanda muito tempo.
▪ Detectar potenciais desencontros entre a arquitetura e os requisitos. Em particular, a avaliação pode examinar alguns aspectos freqüentemente negligenciados nas áreas de operação, administração e manutenção. Como o sistema é instalado? Atualizado? Como nós realizamos a transição das atuais bases de dados?
▪ Avaliar uma ou mais Qualidades arquiteturais específicas: desempenho, confiabilidade, manutenibilidade, Segurança de acesso, segurança de dados e transações.
▪ Identificar oportunidades de reuso.
▪ Testar a arquitetura.</td></tr>
<tr><td colspan="2">Passos
▪ Planejar a Revisão
▪ Preparar a Revisão
▪ Conduzir a Revisão
▪ Alocar as Responsabilidade de Resolução de Defeitos
▪ Testar a Arquitetura</td></tr>
<tr><td>Artefatos de Entrada:
▪ Documento de Arquitetura de Software
▪ Especificações Suplementares
▪ Guidelines de Projeto
▪ Lista de Riscos
▪ Guidelines de Projeto para a Web</td><td>Artefatos de Entrada:
▪ Documento de Arquitetura de Software (aprovado), ou
▪ Requisições de Mudanças (opcional)</td></tr>
<tr><td colspan="2">Responsável: Revisor da Arquitetura</td></tr>
<tr><td colspan="2">Participantes: Arquiteto, Projetista</td></tr>
<tr><td colspan="2">Guidelines de Trabalho:
▪ Conduza a revisão através de um encontro formal, apesar de que os participantes podem preparar algumas revisões por conta própria.
▪ Monitore continuamente a qualidade durante o projeto para evitar um grande número de defeitos que permanecerão até a revisão. Em cada atividade de projeto, os checkpoints listados abaixo podem ser utilizados; use-os para revisões informais no trabalho do dia-a-dia.
▪ Checklists: Arquitetura de Software (geral), Visão de Processo, Visão de Implantação, Modelo de Projeto.</td></tr>
</table>

O Checklist: Arquitetura de Software Baseado na Web contém pontos de verificação da arquitetura específicos para sistemas baseados na Web.

O passo "Testar a arquitetura" só deve ser realizado caso o revisor achar que é propício e necessário. Sistemas baseados na Web compostos somente de páginas de informação não necessitam de ter sua arquitetura testada. O revisor pode alocar a responsabilidade do teste da arquitetura a uma pessoa ou a um grupo de desenvolvedores ou de testadores.

Visto de 20,000 pés não há muito de diferente entre a avaliação da arquitetura de software e qualquer outra avaliação ou revisão.

Contudo, uma característica importante da arquitetura de software é a falta de medidas específicas para muitos atributos de qualidades arquiteturais — somente umas poucas qualidades arquiteturais podem ser objetivamente medidas. Desempenho é um exemplo onde a medida é possível. Outras qualidades são muito mais subjetivas como a integridade conceitual, por exemplo. Contudo, é freqüentemente difícil decidir o que uma métrica significa na ausência de outros dados ou referências para comparação. Se uma referência para o sistema é avaliada e entendida pelo público alvo, é comum expressar alguns dos resultados em uma visão relativa a este sistema referência.

1. No fim da fase de concepção em um ciclo de desenvolvimento inicial, há muito pouco de concreto sobre a arquitetura. Mas uma revisão pode descobrir objetivos não realísticos, partes que estão faltando, oportunidades de reuso, etc.
2. O lugar mais natural para uma avaliação da arquitetura de software é no fim da fase de elaboração. Esta fase foca-se principalmente na exploração dos requisitos em detalhes e na construção da linha base da arquitetura. Neste caso uma grande variedade de qualidades arquiteturais são examinadas.
3. Uma avaliação mais focada pode ser realizada durante a fase de construção para examinar atributos de qualidade específicos, tais como desempenho ou segurança.
4. Avaliação de controle de prejuízos pode ser feita no final da fase de construção ou mesmo na fase de transição, quando certas coisas podem realmente ter dado errado: construção não completa, ou um nível inaceitável de problemas na base instalada durante a transição.
5. Finalmente uma revisão pode ser realizada no fim da fase de transição para inventariar partes reusáveis para eventuais novos produtos ou ciclos de evolução.

Passo: Planejar a Revisão

Propósito
▪ Determinar o foco da revisão
▪ Determinar o escopo da revisão

Antes da revisão, defina o escopo da mesma, determinando as questões que serão revisadas; defina o que será avaliado e porque será avaliado. Veja os checkpoints referenciados acima para os tipos de preocupações que deveríamos ter. As questões exatas dependerão da fase em que se está.

Uma vez que o escopo foi determinado, defina os participantes da revisão, a agenda e a informação que será necessária para realizar a revisão. Tendo selecionados os participantes, estabeleça um balanço entre os conhecimentos da arquitetura de software e os conhecimentos do domínio. Designe um líder para a avaliação.

Revisores deveriam ser experientes na arquitetura de software ou no domínio do problema do sistema em questão.

Passo: Planejar a Revisão

Propósito
▪ Reunir e distribuir material de preparação para a revisão antes das sessões de revisão, de forma que os revisores tenham tempo suficiente para entender a arquitetura e possam formar comentários.
Guidelines: Arquitetura de Software (geral), Visão de Processo, Visão de Implantação, Modelo de Projeto.

A fonte primária de informação para a revisão é o Documento de Arquitetura de Software, suplementado com detalhes adicionais de partes de interesse da arquitetura, do modelo de projeto, notas de projeto ou documentação de explanação adicional. Em adição, checklists de revisão deveriam circular para estimular preocupações e levantar problemas.

Revisores deveriam estudar a documentação, formular preocupações e identificar questões para discussão, antes da revisão.

Passo: Conduzir a Revisão

Propósito
▪ Avaliar a Arquitetura de Software
▪ Identificar problemas na arquitetura.

Em geral, o processo de revisão segue um ciclo repetitivo:

- Uma questão é levantada pelo revisor
- A questão é discutida, e potencialmente confirmada
- Um defeito é identificado
- Continua-se até que nenhuma outra questão seja identificada

Diversas técnicas podem ser usadas para a revisão:

- dirigida à representação
- dirigida à informação
- dirigida à cenário

Revisão dirigida à representação

Obtém-se (ou constrói-se) uma representação da arquitetura, e então levanta-se questões baseando-se nessa representação.

Revisão dirigida à informação

Estabeleça a lista de informações - dados, medidas - que são necessárias, pegue a informação e compare-a aos requisitos ou algum padrão de referência aceitável. Isto aplica-se bem para a investigação de certos atributos de qualidade, tais como desempenho ou robustez.

Revisão dirigida à cenário

Esta é uma técnica sistemática. Transforme as questões gerais a serem levantadas em um conjunto de cenários e percorra estes cenários.

Passo: Alocar as Responsabilidades de Resolução de Defeitos

Propósito
▪ Realizar ações sobre os defeitos identificados.

Depois da revisão, aloque as responsabilidades para cada defeito encontrado. "Responsabilidade" em muitos casos pode não ser consertar o defeito, mas coordenar a investigação de alternativas ou coordenar a resolução do defeito.

Passo: Testar Arquitetura

Propósito

- Testar a arquitetura projetada para o sistema.

Sistemas baseados na Web geralmente possuem arquiteturas robustas, flexíveis e escaláveis, projetadas para providenciar desempenho e para manusear uma carga de transações alta e imprevisível. Estas arquiteturas geralmente são multi-camadas, encapsulando lógica de negócios e integrando múltiplos sistemas legados.

Estas arquiteturas necessitam ser testadas buscando o alcance dos requisitos de desempenho e confiabilidade da aplicação. Quanto mais adiantado estiver o desenvolvimento da aplicação mais caro será a mudança da arquitetura da mesma, caso um problema seja detectado. Dessa forma devemos testar a arquitetura o quanto antes possível. Testando a arquitetura durante o projeto da aplicação, garantimos que nenhum esforço de implementação será despendido inutilmente para a sua construção e que os requisitos de desempenho e confiabilidade serão alcançados.

O teste da arquitetura inicia durante a fase de Elaboração, quando vários protótipos executáveis da arquitetura devem ser construídos. Testar estes protótipos o quanto antes no processo de desenvolvimento permite-nos alcançar um máximo de desempenho e confiabilidade antes que o resto do sistema seja projetado e desenvolvido. Isto permite a produção de uma arquitetura mais estável, evitando que grandes mudanças sejam feitas no sistema em virtude de alterações na arquitetura.

Todos os componentes da arquitetura devem ser testados para evitar que gargalos de desempenho ocorram quando o sistema baseado na Web for implantado.

Os fabricantes geralmente testam o desempenho de seus componentes. Contudo, os fabricantes não testam os padrões de comportamento específico que ocorrem quando seus componentes são invocados por outras aplicações. Também não testam a carga produzida sobre a rede devido a implantação do componente. Se sua arquitetura usa componentes adquiridos de outros fabricantes será necessário testar como estes componentes serão usados pela sua aplicação.

A atividade de teste da arquitetura deverá ser, quando possível, automatizada por alguma ferramenta CASE, como, por exemplo, Rational PerformanceArchitect e Rational LoadTest.

Todos os problemas (desempenho, confiabilidade, etc.) encontrados na arquitetura devem ser repassados para o arquiteto de software, ou para a equipe de arquitetura, responsável.

Atividade: Projetar Subsistema

<table>
<tr><td colspan="2">Propósito
▪ Definir o comportamento especificado nas interfaces dos subsistemas em termos de colaborações de classes contidas.
▪ Documentar a estrutura interna do subsistema.
▪ Definir realizações entre as interfaces dos subsistemas e as classes contidas.
▪ Determinar as dependências entre os subsistemas.</td></tr>
<tr><td colspan="2">Passos
▪ Distribuir o Comportamento do Subsistema para os Elementos do Subsistema
▪ Documentar os Elementos do Subsistema
▪ Descrever as Dependências do Subsistema
▪ Checkpoints: Subsistemas de Projeto</td></tr>
<tr><td>Artefatos de Entrada:
▪ Realizações de Use Case
▪ Subsistemas de Projeto com suas Definições de Interface</td><td>Artefatos Resultantes:
▪ Realizações de Use Case
▪ Subsistemas de Projeto com suas Definições de Interface
▪ Classes de Projeto</td></tr>
<tr><td colspan="2">Freqüência: Uma vez por subsistema de projeto</td></tr>
<tr><td colspan="2">Responsável: Projetista</td></tr>
<tr><td colspan="2">Guidelines: Subsistemas de Projeto, Interfaces.</td></tr>
</table>

Os passos, bem como o detalhamento desta atividade podem ser encontrados na versão 5.0 build 33 (1998) do *Rational Unified Process*.

Atividade: Projetar Classe

<table>
<tr><td colspan="2">Propósito
▪ Garantir que as classes providenciam o comportamento que as realizações de use Case requerem.
▪ Garantir que é fácil implementar as classes.
▪ Manusear os requisitos não funcionais relacionados às classes.
▪ Incorporar os mecanismos de projeto usados pelas classes.</td></tr>
<tr><td colspan="2">Passos
▪ Criar as Classes de Projeto Iniciais
  ▪ Projetar as Classes Limite
  ▪ Projetar as Classes Entidade
  ▪ Projetar as Classes Controle
▪ Identificar as Classes Persistentes
▪ Definir a Visibilidade das Classes
▪ Definir as Operações
  ▪ Identificar as Operações
  ▪ Nomear e Descrever as Operações
  ▪ Definir a Visibilidade das Operações
  ▪ Definir as Operações da Classe
▪ Definir os Métodos
▪ Definir os Estados
▪ Definir os Atributos
▪ Definir as Dependências
▪ Definir as Associações
  ▪ Definir as Associações e Agregações
  ▪ Manusear Associações Subscribes entre Classes de Análise
▪ Definir Generalizações
▪ Manusear Requisitos Não Funcionais em Geral
▪ Avaliar os Resultados</td></tr>
<tr><td>Artefatos de Entrada:
▪ Especificações Suplementares
▪ Guidelines de Projeto
▪ Classes de Análise
▪ Use Cases
▪ Realizações de Use Case
▪ Modelo de Projeto</td><td>Artefatos Resultantes:
▪ Classes de Projeto</td></tr>
<tr><td colspan="2">Responsável: Projetista</td></tr>
<tr><td colspan="2">Mentor de Ferramenta: Usando o Rational Rose para Gerenciar Classes</td></tr>
</table>

Os passos, bem como o detalhamento desta atividade podem ser encontrados na versão 5.0 build 33 (1998) do *Rational Unified Process*.

Atividade: Projetar Use-Case

<table>
<tr><td colspan="2">Propósito
▪ Refinar as realizações de use case em termos de interações.
▪ Refinar os requisitos sobre as operações das classes de projeto.
▪ Refinar os requisitos sobre as operações dos subsistemas e/ou suas interfaces.</td></tr>
<tr><td colspan="2">Passos
▪ Descrever as Interações entre os Objetos de Projeto
▪ Simplificar os Diagramas de Seqüência usando Subsistemas (opcional)
▪ Descrever o Comportamento relacionado à Persistência
▪ Escrevendo Objetos Persistentes
▪ Lendo Objetos Persistentes
▪ Deletando Objetos Persistentes
▪ Modelar Transações
▪ Manusear Condições de Erro
▪ Manusear Controle de Concorrência
▪ Refinar o Fluxo de Descrição de Eventos
▪ Unificar Classes e Subsistemas
▪ Avaliar os Resultados</td></tr>
<tr><td>Artefatos de Entrada:
▪ Especificações Suplementares
▪ Use Cases
▪ Realizações dos Use Case
▪ Classes de Projeto
▪ Subsistemas de Projeto
▪ Interfaces</td><td>Artefatos Resultantes:
▪ Realizações dos Use Case, descritos com diagramas de seqüência, e fluxo de eventos refinado.</td></tr>
<tr><td colspan="2">Responsável: Projetista</td></tr>
</table>

Os passos, bem como o detalhamento desta atividade podem ser encontrados na versão 5.0 build 33 (1998) do *Rational Unified Process*.

Atividade: Projetar Conteúdo

Propósito ▪ Levantar as informações a serem disseminadas. ▪ Editar as informações levantadas.	
Passos	
Artefatos de Entrada: ▪ Guidelines de Projeto para a Web ▪ Modelo de Navegação ▪ Modelo de Use Case ▪ Especificações Suplementares	**Artefatos Resultantes**: ▪ Modelo de Conteúdo
Responsável: Projetista de Conteúdo	

Os passos, bem como o detalhamento desta atividade, não foram definidos neste framework.

Atividade: Projetar Navegação

Propósito ▪ Projetar a forma através da qual os usuários navegarão. ▪ Definir as estruturas navegacionais.	
Passos	
Artefatos de Entrada: ▪ Guidelines de Projeto para a Web ▪ Especificações Suplementares ▪ Modelo de Use Case ▪ Modelo de Análise ▪ Modelo de Projeto	**Artefatos Resultantes**: ▪ Modelo de Navegação
Responsável: Projetista de Navegação	

Os passos, bem como o detalhamento desta atividade, não foram definidos neste framework.

Atividade: Projetar Interface

Propósito	
▪ Definir a aparência dos elementos navegacionais. ▪ Definir quais objetos da interface ativaram a navegação e outras funcionalidades. ▪ Definir quais transformações na interface ocorreram.	
Passos	
Artefatos de Entrada: ▪ Guidelines de Projeto para a Web ▪ Modelo Navegacional	**Artefatos Resultantes**: ▪ Modelo de Interface
Responsável: Projetista de Navegação	

Os passos, bem como o detalhamento desta atividade, não foram definidos neste framework.

Atividade: Projetar Banco de Dados

Propósito	
▪ Garantir que os dados persistentes são armazenados consistente e eficientemente. ▪ Definir o comportamento que deve ser implementado no banco de dados.	
Passos ▪ Mapear as Classes de Projeto Persistentes para o Modelo de Dados ▪ Otimizar o Desempenho do Modelo de Dados ▪ Otimizar o Acesso aos Dados ▪ Definir as Características de Armazenamento ▪ Definir as Tabelas de Referência ▪ Definir Dados e Regras de Entidade Referencial ▪ Distribuir o Comportamento das Classes para o Banco de Dados ▪ Revisar os Resultados	
Artefatos de Entrada: ▪ Especificações Suplementares ▪ Guidelines de Projeto ▪ Modelo de Projeto ▪ Realizações de Use Case ▪ Modelo de Dados	**Artefatos Resultantes**: ▪ Modelo de Dados ▪ Requisições de Mudança no Modelo de Projeto
Responsável: Projetista de Banco de Dados	

Os passos, bem como o detalhamento desta atividade, podem ser encontrados na versão 5.0 *build* 33 (1998) do *Rational Unified Process*.

Guidelines de Projeto para a Web

Objetivos

Uma breve descrição do propósito das Guidelines de projeto para a Web.

Escopo

Uma breve descrição sobre o que as Guidelines de Projeto para a Web se aplicam; o que é afetado ou influenciado por este documento. Outras seções podem ser inclusas neste documento.

Referências

Uma lista de documentos relacionados ou referenciados.

Guidelines Gerais de Projeto e Implementação para a Web

Esta seção descreve os princípios e estratégias a serem usadas durante o projeto e implementação do sistema baseado na Web.

Convenções de Modelagem

1. Criar diagrama de componentes, para a visão de implantação, especificando os vários pacotes que compõem a aplicação(componentes de software e páginas de informação), mostrando a localização desses componentes nas respectivas camadas.
2. Representar a visão de implantação através de diagramas de implantação ressaltando todas as características dos nós envolvidos na arquitetura da aplicação para a Web, o tipo de *gateway* empregado, o tipo de *middleware* que será necessário, o tipo da plataforma de computação, etc.
3. Todas as abstrações chave e classes de análise levantadas durante a análise arquitetural e a análise de use case que representem páginas de informação devem ser identificadas pelo estereótipo *boundary* e pelo mecanismo de análise "HTML".
4. A partir do projeto da arquitetura as páginas de informação, cujo propósito é a apresentação de informações, devem ser modeladas segundo a seguinte padrão:

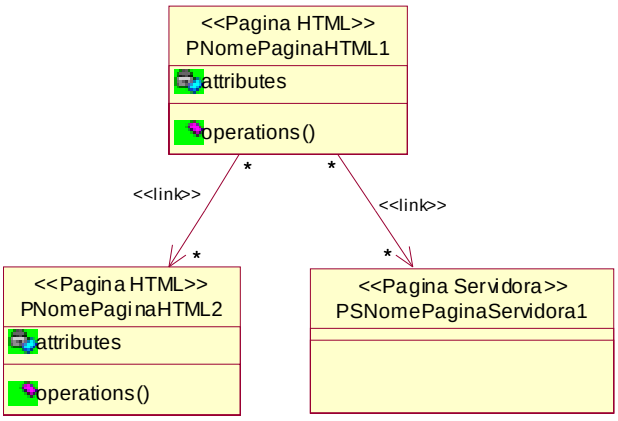
Padrão Pagina HTML

- **Nome:** Pagina HTML

- **Objetivo:** Representar as páginas de informações cujo propósito é a apresentação de informações. Estas páginas não interagem com nenhum componente de software.

- **Motivação:**
  1. Separação da representação das páginas de informação dos componentes de software.
  2. Separação da representação dos tipos de páginas de informação: páginas que somente apresentam informações e páginas que executam funções de componentes de software.

- **Estrutura:**



1. Uma pagina HTML pode estar ligada unicamente a outra pagina HTML ou a uma página que execute alguma função de um componente de software, aqui representada pelo padrão Pagina Servidora.
2. Os atributos de uma Pagina HTML representam as informações que a mesma deve apresentar. Exemplo: Supondo que a PNomePaginaHTML1 fosse a página principal de um site de compras; compras de cd, compras de livros, compras de produtos eletrônicos, logotipo do site, banner para promoções e seriam possíveis atributos para esta página.
3. As operações de uma Pagina HTML representam todas as funções que eventualmente podem ser acionadas através da mesma, inclusive ligações (*links*) com outras páginas. Exemplo: Supondo que a PNomePaginaHTML1 fosse a página principal de um site de compras; acessar página de cd, acessar página de livros e acessar página de promoções seriam possíveis operações para esta página.

- **Colaborações:** Uma Pagina HTML pode estar ligada a várias outras Paginas HTML ou a páginas de informação que possuam associadas a elas algum processamento baseado no servidor. A partir de uma Pagina HTML pode se navegar somente para outras páginas de informação.

- **Aplicabilidade e Conseqüências:**

1. Use o padrão quando você quiser representar páginas de informações, suas informações e suas operações.
2. Facilita a legibilidade dos diagramas que descrevem a estrutura da aplicação.
3. Identifica classes que farão parte do modelo de navegação.

Mecanismos de Análise

1. Mecanismos de análise capturam aspectos chaves de uma solução de forma independente da implementação. É fundamental em sistemas para a Web a diferenciação entre páginas de informação e componentes de aplicação. Desta forma as abstrações chave e as classes de análise que representarem páginas de informação deverão ser identificadas através do mecanismo de análise “HTML”. Durante o projeto arquitetural quando os elementos de análise estiverem sendo transformados em elementos de projeto, este mecanismo de análise poderá ser mapeado utilizando-se a convenção de modelagem "Pagina HTML" e o padrão de projeto "Pagina Servidora" descritos neste documento. O propósito único deste mecanismo é a identificação das classes que representam as páginas de informação.

Reusabilidade de Componentes

1. Evite a *overengineering*. Sempre que possível use componentes já criados. Aplicações para a Web devem ser desenvolvidas rapidamente para serem competitivas. Fique de olho no Web Time ou Internet Time.
2. É melhor comprar do que desenvolver. Isto é especialmente verdade no caso de software baseado na Web. Uma grande quantidade de esforço e dinheiro pode ser economizado. Preste atenção no *freeware*.

Guidelines para o Projeto de Banco de Dados

Esta seção define regras e recomendações para o projeto do banco de dados para um sistema baseado na Web.

1. Determine quais as fontes de dados (HTML, arquivos, bancos de dados) serão necessárias, onde elas estão localizadas e como acessá-las.
2. Não distribua demais os dados (*overdistribute*). O Compartilhamento de dados altamente distribuídos entre milhares de usuários torna-se mais difícil e o desempenho pode piorar, uma vez que a rede pode possuir mais pontos de congestionamento.
3. Use a visão de implantação para descrever a distribuição física dos dados através dos nós.

Guidelines para a Definição da Arquitetura

Esta seção define regras e recomendações para o projeto arquitetural de um sistema baseado na Web.

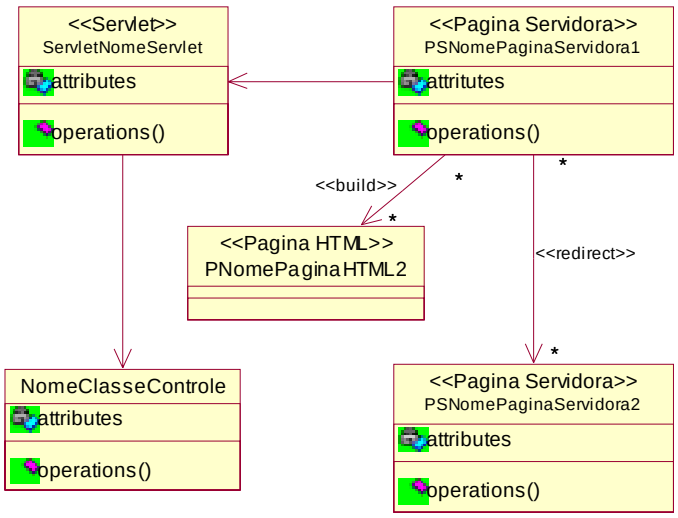
1. Determine o que será escrito como aplicação e o que será escrito como páginas de informação.
2. Analise a porção de código que será processada nas várias camadas da aplicação (cliente, Servidor Web, Servidor de Dados, Servidor de Aplicação, etc ...) de forma que não haja sobrecarga de processamento.
3. Verifique se a arquitetura está adequada quanto ao desempenho e à confiabilidade. Técnicas de redundância de dados e de processamento podem ser usadas para tal fim.
4. Interações entre dados e programas através de múltiplos computadores e sistemas na Internet introduzem muitos problemas de segurança e controle de integridade. A arquitetura deve ser projetada para permitir o máximo de segurança tanto a nível de rede (firewalls que filtram o tráfico não permitido e servidores proxy que reescrevem pacotes para esconder sua fonte original) e a nível de aplicação (autenticação, autorização e segurança de controle, de dados e de transação).
5. Uma aplicação em um site pode afetar muitas aplicações que residem em outros sites. Desta forma muitos pontos de falhas são introduzidos. Interfaces com sistemas legados e outros sistemas na Web só devem existir se forem realmente necessárias e devem ser cercadas de todo tipo de segurança possível.
6. Verifique se os requisitos operacionais foram corretamente elicitados pois representam uma grande quantidade de informação, de desempenho, segurança e interconectividade. Projete a arquitetura e o sistema para atender estes requisitos.
    - Requisitos de escala e crescimento (número de usuários que acessarão o sistema, quantidade de operações realizadas pelos mesmos, freqüência de chegada de transações em diferentes sites, tamanho do banco de dados).
    - Requisitos de tempo de resposta dos usuários (média, pior caso).
    - Requisitos de segurança (criptografia, id seguro).
    - Restrições de disponibilidade (7 dias por semana, 24 horas por dia).
    - Conformidade com os padrões de *middleware* existentes (CORBA, HTTP, CGI, SMTP, RMI).
    - Necessidade de conectividade de usuários de várias plataformas (PCs, Macs, usuários Unix conectados sobre TCP/IP LAN, redes de pacotes ou linhas dial-up).
    - Interoperabilidade e interface com outros sistemas.
    - Backup/recuperação necessárias e manuseio de desastres.
    - Restrições (padrões organizacionais e políticos a serem seguidos, controle de aplicação e restrições de auditoria).

7. Esconda os detalhes irrelevantes da arquitetura para reduzir a complexidade e exponha os detalhes necessários para obter controle.
8. Inicie a definição da arquitetura do sistema assumindo que todos os componentes são fisicamente ou logicamente remotos uns dos outros.
9. Projete a arquitetura para que ela tolere todos os tipos de falhas, incluindo falhas parciais em um ou outro componente.
10. Componentes arquiteturais devem ser eficientes em sistemas pequenos e suficientemente funcionais em grandes sistemas.
11. Não distribua demais (*overdistribute*). Se dois processos interagem freqüentemente, então ponha-os na mesma máquina.

Padrões de Projeto

Pagina Servidora

- **Nome:** Pagina Servidora
- **Objetivo:** Representar as páginas de informações que acionam funções de componentes de software residentes no servidor. Estas páginas se comunicarão a estes componentes de software através de servlets.
- **Motivação:**
  1. Separação da representação das páginas de informação dos componentes de software.
  2. Separação da representação dos tipos de páginas de informação: páginas que somente apresentam informações e páginas que acionam funções de componentes de software.
  3. Representação de páginas de informação que apresentam algum processamento associado.
- **Estrutura:**



1. Uma Pagina Servidora pode estar ligada unicamente a outra Pagina Servidora ou a uma página que apresente informações, aqui representada pelo padrão Pagina HTML.
2. Os atributos de uma Pagina Servidora representam as informações que a mesma deve apresentar. Exemplo: Supondo que a PNomePaginaServidora1 fosse uma página para a realização de compras de cds; nome do cd, quantidade e local de entrega seriam possíveis atributos para esta página.
3. As funções de uma Pagina HTML representam todas as operações que eventualmente podem ser acionadas através da mesma, inclusive ligações (*links*) com outras páginas. Exemplo: Supondo que a PNomePaginaServidora1 fosse a página para a realização de compras de cds; calcular preço do transporte e confirmar compra seriam possíveis operações para esta página.

- **Colaborações:** Uma Pagina Servidora delega responsabilidades para uma Classe de Controle (componente de software da aplicação) através de um Servlet. Toda Pagina Servidora pode se ligar a várias outras Paginas Servidoras ou a páginas de informação que somente apresentam informações.
- **Aplicabilidade e Conseqüências:**
  1. Use o padrão quando você quiser representar páginas que apresentam processamento associado no servidor, suas informações e suas ligações.
  2. Facilita a legibilidade dos diagramas que descrevem a estrutura da aplicação, identificando a ligação entre as páginas de informação e os componentes de software.
  3. Identifica classes que farão parte do modelo de navegação.

Recomendações para o Desenvolvimento de Sistemas baseados na Web

1. Use o princípio KISS (*Keep it Simple Sir*):
   - Se possível inicie com um único fornecedor de *middleware*
   - Limite o *mix-and-matching*
   - Não distribua demais sua aplicação.
2. A divisão de uma grande aplicação em componentes, subsistemas e módulos não é uma tarefa fácil. Ainda mais quando cada pedaço reside em um computador em separado. Desta forma, muitas configurações de aplicações e opções de interconectividade não são cuidadosamente avaliadas.
3. Uma tendência das aplicações atuais é a separação entre regras de negócio, apresentação (interface do usuário) e gerenciamento dos dados da aplicação. Esta separação de preocupações leva a uma arquitetura de aplicação em três camadas bastante apropriada para sistemas baseados na Web, pois promove a noção de interoperabilidade, reuso e gerenciamento das aplicações em ambiente distribuído.

Checkpoints: Arquitetura de Software baseado na Web

Geral

- As Guidelines de Projeto para a Web foram seguidas?
- Os componentes a serem reusados foram totalmente encontrados?
- Os componentes e sua distribuição nas diversas camadas foram representados em um diagrama de componentes?
- Todos os detalhes da arquitetura do sistema (infraestrutura) foram representados no diagrama de implantação?
- As fontes de dados e sua distribuição física através dos nós foram representadas no diagrama de implantação?
- Os componentes de aplicação e as páginas de informação foram determinados?
- O processamento foi distribuído corretamente (entre clientes, servidores de dados, servidor Web, servidores de aplicação, etc.) de forma a não haver sobrecarga de processamento?
- A arquitetura está adequada quanto ao desempenho e a confiabilidade?
- As interfaces com sistemas legados e outros sistemas na Web são realmente necessárias?
- Os detalhes irrelevantes da arquitetura foram escondidos para diminuir a complexidade?
- Os componentes arquiteturais são eficientes sobre carga pequena e funcionais sobre uma carga alta de processamento?
- O sistema não foi distribuído demais?

Segurança

- Mecanismos de autenticação e autorização foram projetados?
- A aplicação está segura do ponto de vista da rede? Você projetou os *firewalls* e os servidores *proxy* necessários?

Checkpoints: Análise Arquitetural

Para Sistemas baseados na Web:

- As abstrações chaves (classes de análise) foram identificadas corretamente com o mecanismo de análise “HTML”?
- As abstrações chaves (classes de análise) modelam corretamente e suficientemente o domínio do problema da aplicação para a Web em desenvolvimento?
- O Artefato: Guidelines de projeto para a Web está sendo atualizado corretamente?
- Os conceitos de "Definição de Camadas" e "Padrões de Distribuição" foram observados durante a estruturação inicial da aplicação?
- A estrutura inicial do Web site foi levantada corretamente?

Conceito: Princípios para o Reuso de Componentes

Em experiências relatadas por muitas empresas com o reuso foi encontrado um número de princípios que são comuns à maioria delas. Para que o reuso sistemático de software seja alcançado deve-se ter em mente este conjunto de princípios:

1. Planeje e adapte a arquitetura do sistema, o processo de desenvolvimento e a organização para as necessidades de reuso em uma visão sistemática mais incremental. Inicie com projetos pilotos e então, aumente a escala.
2. Planeje para iniciar o reuso com a arquitetura e um processo incremental de definição arquitetural.
3. Crie e desenvolva componentes reusáveis em um ambiente de trabalho real.
4. Gerencie sistemas de aplicação e componentes reusáveis como um portfólio de produto de valor financeiro, focando o reuso sobre componentes comuns em domínios de subsistema.
5. Idealize que um objeto ou componente tecnológico isolado não é suficiente.
6. Invista na incrementação continua da infra-estrutura, educação de reuso e experiência.
7. Meça o progresso do reuso com métricas, e otimize o programa de reuso.