Patrícia Marques Rodrigues de Souza Álvares
pmarques@dcc.ufmg.br

WebPraxis - Um processo personalizado para projetos de desenvolvimento para a Web

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

27 de março de 2001

Agradecimentos

Agradeço a Deus, por me dar forças para seguir em frente.

Agradeço aos meus pais e a minha irmã pelo apoio, amizade, e por compreenderem minhas ausências.

Agradeço a Fá, pelo incentivo constante; a Junsky e a Gigi pela troca de idéias; a Rê, por estar sempre ao meu lado e pela paciência em revisar todo o texto.

Agradeço à Cristiane pela ajuda na revisão do texto.

Agradeço a todos meus amigos e minha família, pois todos, de alguma forma, contribuíram para a minha conquista.

Agradeço ao professor Wilson de Pádua Paula Filho pela orientação, pela troca de idéias, pelo incentivo e pela revisão de todo o trabalho.

Agradeço também a todos os que foram meus professores no DCC e que possibilitaram minha formação em Ciência da Computação. Agradeço, em especial, aos professores Márcio Luiz Bunte de Carvalho, pelo apoio e amizade desde o curso de graduação; ao professores Roberto da Silva Bigonha e João Eduardo de Rezende Dantas, pelo apoio inicial no curso de pós-graduação; à professora Mariza Andrade Silva Bigonha, com quem trabalhei durante muito tempo, pela dedicação, incentivo e amizade; ao professor Clarindo Isaías Pereira da Silva e Pádua, por permitir que utilizasse o laboratório do grupo de Usabilidade.

Agradeço a todos os funcionários do DCC, que trabalham para manter funcionando toda a estrutura necessária para a formação dos alunos e, em especial, à Renata Viana Moraes Rocha, pela dedicação, competência e disposição.

Agradeço ao CNPQ e CAPES por financiarem parte desse projeto e à BHS, à BMS e ao convênio DCC – Tribunal de Contas por possibilitarem sua finalização.

Agradeço especialmente à DoctorSys e a todos que dela fizeram (ou ainda fazem) parte, pelo incentivo inicial, apoio e amizade.

Resumo

A adoção de um processo definido e adaptado para a realidade de uma organização produtora de software é um fator decisivo para a geração de sistemas de alta qualidade que atinjam as necessidades dos usuários finais e sejam feitos dentro de um cronograma e orçamento previsíveis. Este trabalho apresenta uma personalização para o processo padrão Praxis, direcionada para projetos de desenvolvimento para a Web. O Praxis é um processo completo de desenvolvimento de software adequado para projetos com duração de seis meses a um ano, em um contexto educacional ou de treinamento, cobrindo tanto métodos técnicos (requisitos, análise, desenho, implementação e testes) quanto gerenciais (gestão de compromissos, gestão de projetos, gestão da qualidade e engenharia de processos). O novo processo gerado – WebPraxis – se destina a projetos pequenos, direcionados para a Web, com duração máxima de um semestre e com um grupo de um a quatro desenvolvedores. A notação adotada pelo WebPraxis utiliza uma extensão da UML, conhecida por WAE (Web Application Extension for UML), que traz novos estereótipos para modelar aspetos específicos do mundo Web. O processo traz diretrizes e enfatiza aspectos importantes a serem observados pelos analistas em cada fase de um projeto de desenvolvimento para a Web. Sua definição foi realizada com base na metodologia para definição de processos proposta em [Humphrey95]. Visando verificar se o processo criado é factível, realizou-se um estudo de caso, utilizando-o no desenvolvimento de uma pequena aplicação Web. Uma vez definido o processo, este deve estar em constante aperfeiçoamento para incluir tecnologias que vão surgindo e corrigir possíveis falhas detectadas com sua aplicação em projetos reais.

Abstract

The adoption of a defined process by a software development organization is a decisive factor in the production of high quality systems that meet the user's needs and follow a predictable schedule and budget. This work shows a specialization for the Praxis process, aimed at Web development projects. Praxis is a complete software development process adequate to projects that take from six months to one year, covering both technical methods (requirements, analysis, design, implementation and tests) and management methods (contract, project and quality management and project engineering). The new process presented here – WebPraxis – is destined for small projects, involving Web development, which take from six months to one year and with a team from one to four developers. The notation adopted by WebPraxis uses an extension of UML named WEA (Web Application Extension for UML), which includes new stereotypes for modeling web-specific architectural elements. The process brings directives and emphasizes important aspects that practitioners must evaluate in each phase of a Web development project. Its definition was based on the methodology for process definition proposed by [Humphrey95]. Aiming at verifying if the new process works in practice, it was applied on the development of a small web application. Once the process is defined, it has to be improved constantly to include new technologies and to correct possible flaws detected by its application to a larger number of projects.

Sumário

Lista de figuras

Lista de tabelas

Capítulo 1

Introdução

1.1 Motivação

Uma organização produtora de software bem sucedida é aquela que desenvolve de forma consistente sistemas de qualidade que atendam às necessidades de seus futuros usuários. Uma organização que é capaz de desenvolver tal software com um prazo e custo previsíveis, utilizando de forma eficiente seus recursos tanto humanos quanto materiais, está à frente da grande maioria de seus concorrentes [Booch98].

Atingir tais objetivos não é tarefa simples e novas metodologias, paradigmas e tecnologias para auxiliar no desenvolvimento de sistemas são alvo de pesquisa constante por parte dos engenheiros de software. Apesar de não existir uma fórmula mágica que resolva todos os problemas e que sirva para o desenvolvimento de qualquer tipo de sistema, parece haver um consenso quanto a três aspectos que precisam estar presentes para que um projeto de desenvolvimento seja bem sucedido: um processo bem definido, uma notação e ferramentas que automatizem as partes mais repetitivas e complexas desse processo [Quatrani98].

Um processo e uma notação são partes integrantes de uma metodologia de desenvolvimento. O processo irá guiar os profissionais envolvidos durante todo o ciclo de vida do sistema. O processo é sustentado por vários conceitos e a notação é responsável por expressar estes conceitos [Blaha+98]. Estes dois aspectos – processo e notação – são alvo dos estudos realizados neste trabalho. Quanto às ferramentas de apoio a um processo e de suporte às várias fases de desenvolvimento, direcionamos o leitor para [Carvalho01], para maiores informações, por fugir do escopo deste trabalho.

A notação utilizada junto a um processo de desenvolvimento possibilita a comunicação de forma padronizada e organizada dos aspectos do sistema que são modelados. A modelagem é uma parte crucial da maioria das atividades que levarão à produção de um software de qualidade. Modelos são construídos para explicitar a estrutura e o comportamento esperados do sistema, para visualizar e controlar sua arquitetura, para melhor compreender o sistema que está sendo desenvolvido possibilitando assim simplificações e reuso [Booch98]. Sem uma notação comum não seria possível dois profissionais distintos trabalharem num mesmo modelo, ou mesmo um compreender o que o outro colega desejava expressar ao fazer determinado modelo.

O processo, por sua vez, é a componente metodológica que provê uma abordagem disciplinada para a atribuição de tarefas e responsabilidades no contexto do desenvolvimento de

um sistema. Sua finalidade é garantir a produção de sistemas de alta qualidade, atingindo as necessidades dos usuários finais, dentro de um cronograma e um orçamento previsíveis.

Um processo de software é um conjunto de fases de um projeto, estágios, métodos, técnicas e práticas que são empregadas para desenvolver e manter o software e os artefatos associados a ele, como planos, documentos, modelos, código, casos de teste, manuais, etc [Ambler99b]. Geralmente, um processo se faz necessário quando serão executadas atividades repetitivas como escrever um programa, analisar um requisito, executar um teste, etc.

Segundo [Paula01], a existência de processos definidos[1] é necessária para a maturidade das organizações produtoras de software. Os processos definidos permitem que a organização tenha um *modus operandi* padronizado e reprodutível. Isto facilita a capacitação das pessoas, e torna o funcionamento da organização menos dependente de determinados indivíduos. Entretanto, não é suficiente que os processos sejam definidos. Processos rigorosamente definidos, mas não alinhados com os objetivos da organização são impedimentos burocráticos, e não fatores de produção.

Ainda, para salientar a importância de um processo adequado de desenvolvimento, [Ambler99b] cita como razões pelas quais um processo aumenta a produtividade em uma organização:

- Possibilita uma melhor compreensão de como o software será produzido, e assim pode-se tomar decisões mais acertadas sobre a escolha de ferramentas de apoio e contratação de mão-de-obra.
- Permite minimizar os esforços, promovendo o reuso e uma integração consistente entre diferentes equipes.
- Permite melhorar o aproveitamento dos esforços de manutenção e suporte, definindo estratégias para gerência de melhorias e manutenções (o que deve ser feito imediatamente e o que deve ser deixado para futuras versões) e também estratégias para a operação e suporte.
- Permite melhor gerenciar a complexidade crescente dos sistemas de software. Com o aumento constante da complexidade dos sistemas, faz-se ainda mais necessária uma forma efetiva de desenvolver e manter os mesmos.
- Possibilita gerenciar os vários projetos que sua organização possa ter num dado momento. Na medida em que aumenta o número de projetos, uma forma organizada e padronizada de desenvolvimento faz-se necessária para que seja possível gerenciá-los.
- Possibilita assimilar novas tecnologias de forma menos traumática para a organização.
- Possibilita um cálculo do custo de um projeto mais próximo à realidade.

De todos estes aspectos podemos concluir que se uma organização sempre atua num único projeto de cada vez, a tecnologia envolvida nesses projetos é sempre a mesma, o número de pessoas envolvidas é bastante pequeno, a complexidade dos sistemas é baixa e nunca existem problemas de manutenção, realmente não seria necessário um processo bem definido. Entretanto, no mundo da informática, é quase impossível encontrar uma organização que esteja sobrevivendo

[1] Entenda-se por "processo definido" um processo para o desenvolvimento de software formalizado de alguma maneira, documentado e que possa seguido por todos os desenvolvedores. Algum processo para desenvolver software, ainda que informal, todos têm, mas o alvo deste trabalho são os processos padronizados e definidos para uma organização.

até hoje com todas essas características. Na medida em que aumentam a complexidade dos sistemas, o número de sistemas desenvolvidos simultaneamente (conseqüentemente, o número de pessoas a serem gerenciadas) se torna imprescindível a utilização de uma abordagem mais organizada e estruturada de trabalho. Somando-se a estas variáveis o constante avanço tecnológico e a necessidade de manutenção e melhorias dos softwares já desenvolvidos, a utilização de um processo de desenvolvimento bem definido se torna ainda mais indispensável. É exatamente nesse ponto que um processo bem definido pode ajudar.

O processo que constitui a base deste trabalho é o Praxis [Paula01]. O Praxis é um processo padronizado, criado no DCC-UFMG pelo professor Wilson de Pádua, que integra um conjunto de métodos e padrões cobrindo todos os principais tópicos da Engenharia de Software. Para passar à prática de uma organização produtora de software, o Praxis pode ser complementado e personalizado. Novos padrões podem ser adicionados, que cubram aspectos específicos das aplicações, da tecnologia, dos métodos gerenciais e até da cultura da organização. Pode-se também optar por usar as lições aprendidas através de um processo com suporte comercial, como o Processo Unificado [Jacobson+99] e de outros processos incluídos em ferramentas de Engenharia de Software.

O Praxis é um processo genérico para a produção de aplicativos interativos e orientados a objetos. Uma especialização interessante seria a de adaptar o Praxis para projetos de desenvolvimento para a Web, que se enquadram neste âmbito, mas utilizam tecnologia bem mais específica. Como conseqüência do crescimento do comércio eletrônico e da Web de maneira geral, grande parte das empresas da área de informática já possui projetos voltados para a Internet, ou irá possuir em breve.

Essa nova geração de projetos possui muitas similaridades com o desenvolvimento de software tradicional, mas também difere deste em um número significativo de áreas [Ward+99]. Segundo [Fraternali99], aplicações Web em domínios como comércio eletrônico, bibliotecas digitais e educação a distância são caracterizadas por uma mistura sem precedentes de diferentes áreas, fazendo com que sejam radicalmente diferentes das aplicações da geração anterior. As aplicações Web são acessadas por uma enorme gama de pessoas, de diferentes nacionalidades, costumes, idades, com pouca ou nenhuma habilidade em utilizar um computador, o que gera a necessidade de novas interfaces homem-máquina capazes de capturar a atenção dos usuários e facilitar ao máximo o acesso à informação. Um outro aspecto, também novo, trazido por estas aplicações é a necessidade de integração e gerenciamento de diferentes tipos de dados, estruturados e não estruturados, disponibilizados por diferentes fontes e possivelmente armazenados através de diferentes tecnologias, como bancos de dados, arquivos, dispositivos de armazenagem multimídia, etc.

As aplicações Web podem ser vistas como sistemas híbridos, trazendo características de aplicações hipermídia – encontradas em CD-ROMS, quiosques – e também de sistemas de informação tradicionais [Fraternali99]. Como em uma aplicação hipermídia, a informação é acessada de forma exploratória e a maneira como está sendo apresentada e que pode ser navegada é de grande importância. De forma similar aos sistemas de informação, o tamanho e volatilidade dos dados e a distribuição das aplicações requerem soluções consolidadas quanto a sua arquitetura, baseando-se em tecnologias como sistemas de gerência de bancos de dados e a divisão do sistema em camadas.

Ao se desenvolver uma aplicação para a Web, por um lado o que está sendo criado é um tipo de mídia de consumo, como um vídeo ou um filme. Como qualquer produto de consumo, o *look and feel* de uma aplicação Web é absolutamente crítico e dessa forma, faz-se necessário o envolvimento de um novo grupo de profissionais incluindo pessoas de marketing, designers gráficos e gerentes de negócio. Por outro lado, os sistemas desenvolvidos para a Web devem ser robustos e bem elaborados o suficiente para conseguirem oferecer tudo o que se espera deles: interatividade, flexibilidade, se mostrar amigável[1] e fácil de usar, dar suporte ao processamento de transações *on-line*, ser personalizado, estar disponível 7 dias por semana e 24 horas por dia, ser facilmente modificado e atualizado para acompanhar o dinamismo do mercado, muitas vezes integrar e apresentar de forma mais moderna um conjunto de diferentes sistemas legados, etc. E tudo isso deve ser desenvolvido e disponibilizado o mais rápido possível, pois quando se trata de Web, os prazos são ainda menores e mais críticos se comparados aos de desenvolvimento de sistemas tradicionais.

Para atender a todos esses aspectos citados anteriormente faz-se necessário um processo que abranja tanto as novas áreas introduzidas no escopo do problema do desenvolvimento de um sistema, quanto aquelas similares ao desenvolvimento de uma aplicação tradicional. Assim pode ser definido o WebPraxis: ele herda do Praxis o suporte às áreas comuns entre os dois mundos, especializa algumas partes e insere outras novas, para atender o que uma aplicação Web tem de singular em relação às outras.

1.2 Objetivos e Resultados Esperados

O objetivo deste trabalho é elaborar uma personalização para o processo padrão Praxis, direcionada para projetos de desenvolvimento para a Web. Esta personalização levará em consideração os aspectos específicos de aplicações Web, apontando possíveis soluções e guiando os analistas durante as fases de desenvolvimento deste tipo de sistema.

Após criar o novo processo, espera-se verificar que ele é factível na prática através de sua utilização no desenvolvimento de uma pequena aplicação. Nesta experiência, poderão ser identificados pontos fracos e possíveis melhorias.

Com a elaboração do novo processo, resultante da personalização do Praxis, espera-se também validar a capacidade do próprio Praxis de ser utilizado em projetos reais quando adequado e personalizado para tal.

[1] *User-friendly*.

1.3 Organização deste documento

Este documento foi dividido em sete capítulos que obedecem a ordem em que este trabalho foi desenvolvido:

- **Capítulo 2 - Trabalhos Relacionados**: apresenta, para embasar este trabalho, o estudo dos processos e metodologias de desenvolvimento de software existentes tanto para sistemas de informação tradicionais quanto para sistemas multimídia.

- **Capítulo 3 - Modelagem de aplicações Web**: são apresentados aspectos importantes a serem considerados na modelagem de aplicações Web. Traz também maiores detalhes sobre a notação a ser utilizada pelo WebPraxis, visto que é uma extensão relativamente recente da UML e ainda não de domínio comum.

- **Capítulo 4 - A Definição de um Processo**: neste capítulo é descrita a metodologia para definição de processos utilizada na criação do WebPraxis. São apresentados também vários aspectos importantes que devem ser considerados na definição de um processo e sua posterior aplicação em uma organização.

- **Capítulo 5 - O WebPraxis**: apresenta a personalização do Praxis em si, mostrando todos os passos para a geração do novo processo e em seguida o resultado: o WebPraxis.

- **Capítulo 6 - Estudo de Caso**: neste capítulo é descrito o estudo de caso desenvolvido para verificar se o novo processo é realmente factível.

- **Capítulo 7 - Conclusão e Trabalhos Futuros**: por fim, este capítulo sintetiza as contribuições deste trabalho e apresenta as conclusões. Traz, ainda, possíveis melhorias na presente abordagem e sugere trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Para embasar o presente trabalho, foram analisados tanto processos voltados para o desenvolvimento de sistemas tradicionais, quanto metodologias de desenvolvimento de sistemas hipermídia. O primeiro grupo compreende os trabalhos descritos na seção 2.1. O segundo grupo compreende os trabalhos descritos na seção 2.2.

Após analisar estes trabalhos, pode-se constatar que os fundamentos encontrados em todos eles são os mesmos, pois os fundamentos do desenvolvimento de software não variam e já são conhecidos há muitos anos. É preciso fazer um levantamento de requisitos, é preciso modelar, é preciso codificar, é preciso testar, é preciso fazer o controle de futuras modificações [Ambler00]. Mas cada um destes trabalhos traz também aspectos únicos, de acordo com a finalidade de cada um, de acordo com a que cada um se destina. Sendo assim, todos puderam contribuir um pouco, direta ou indiretamente, para o trabalho e o amadurecimento das idéias nele propostas.

2.1 Desenvolvimento de Sistemas

2.1.1 OPEN

(Object-oriented Process, Environment and Notation) [Graham+97]

Processo criado pelo *OPEN consortium*[1], destina-se principalmente a organizações que utilizam tecnologia de orientação a objetos e componentes, mas pode também ser adaptado e empregado junto a outras tecnologias de desenvolvimento de software [Ambler00]. O OPEN foi criado a partir da junção de alguns métodos já existentes e utilizados anteriormente [Henderson98]. Quanto à notação, podem ser utilizadas a UML, a OML (*Object Modeling Language*) ou qualquer outra notação que consiga documentar os produtos gerados pelo processo, ou seja, ele é independente de notação.

Na Figura 1, pode-se observar o ciclo de vida do OPEN. As atividades mostradas nos retângulos correspondem, de certa forma, aos fluxos do UP. Estas atividades descrevem a arquitetura geral do processo, sendo completadas por tarefas de menor escala que focalizam a parte gerencial do projeto. As atividades e as tarefas descrevem o que precisa ser feito e o como deve ser feito é descrito por um conjunto de técnicas sugeridas.

[1] Grupo de indivíduos e organizações que se destina a promover e fortalecer o uso da orientação a objetos.

Na primeira atividade realizada, a de Iniciação do Projeto, devem ser feitas as adequações necessárias para se aplicar o processo a uma determinada organização e a um determinado projeto. Tal adequação é feita selecionando-se as tarefas e as técnicas apropriadas para o projeto. As atividades seguintes são responsáveis pelo suporte à captura de requisitos, à análise e desenho OO. A construção do sistema é feita de uma maneira evolutiva, como indicam os retângulos sobrepostos no diagrama para essa atividade.

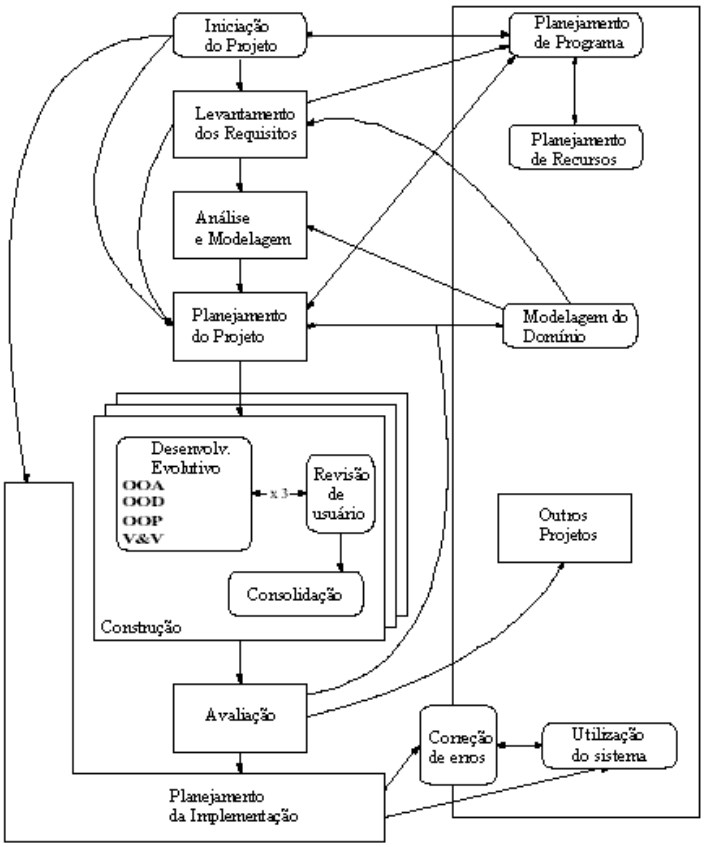

Figura 1 – O ciclo de vida do OPEN

Até aqui, as semelhanças com o UP são grandes. A maior diferença pode ser observada na porção direita do diagrama. As atividades apresentadas na porção esquerda da Figura 1 dizem respeito a um único projeto, enquanto que as apresentadas na direita são atividades cujo escopo não se limita a um único projeto, mas aos vários projetos da organização. Este conjunto de atividades é chamado de Gerência de Programa[1] no OPEN e não aparece na versão atual do UP. Tais atividades são responsáveis por esforços para melhorias no processo, modelagem da arquitetura do projeto e o desenvolvimento/manutenção de padrões e modelos.

Alguns dos pontos fortes deste processo são o suporte à modelagem de processos, à engenharia de requisitos e à gerência de projetos, preocupando-se sempre com a qualidade de software e com o uso de métricas. O OPEN pode ser dirigido por casos de uso, por responsabilidade ou por dados, o que, segundo [Henderson+99], é uma vantagem dele sobre o UP. Outras vantagens seriam não depender de uma notação específica e ter a parte de gerência de projetos (atividades e técnicas) descrita de forma completa.

[1] Programa, neste caso, significando um conjunto de projetos ou um conjunto de versões de uma mesma aplicação.

Segundo [Ambler00], por ser resultado da junção de vários métodos e da experiência de diferentes desenvolvedores e acadêmicos, o OPEN é o mais abrangente dos processos voltados para tecnologia OO. O conjunto de técnicas que ele traz como sugestão de como realizar cada tarefa é vasto o bastante para que se possa encontrar e escolher aquelas que mais se adeqüem a sua realidade.

2.1.2 OOSP

(Object-Oriented Software Process) [Ambler98][Ambler99a]

O OOSP é composto por uma coleção de padrões de processo[1].

Cabe aqui um parêntese sobre padrões de processo. Um padrão de processo é uma coleção de técnicas, ações, tarefas e atividades com a finalidade de resolver um problema específico dentro de um processo de desenvolvimento de software [Ambler99b]. Padrões de processo são uma ótima forma de se disponibilizar abordagens para o desenvolvimento de software que se mostraram eficientes na prática. Uma característica importante de um padrão de processo é que ele descreve o que deve ser feito, sem entrar em detalhes de como deve ser feito. Dessa forma, os padrões podem ser utilizados como blocos a partir dos quais pode-se montar um processo que supra as necessidades de uma organização específica.

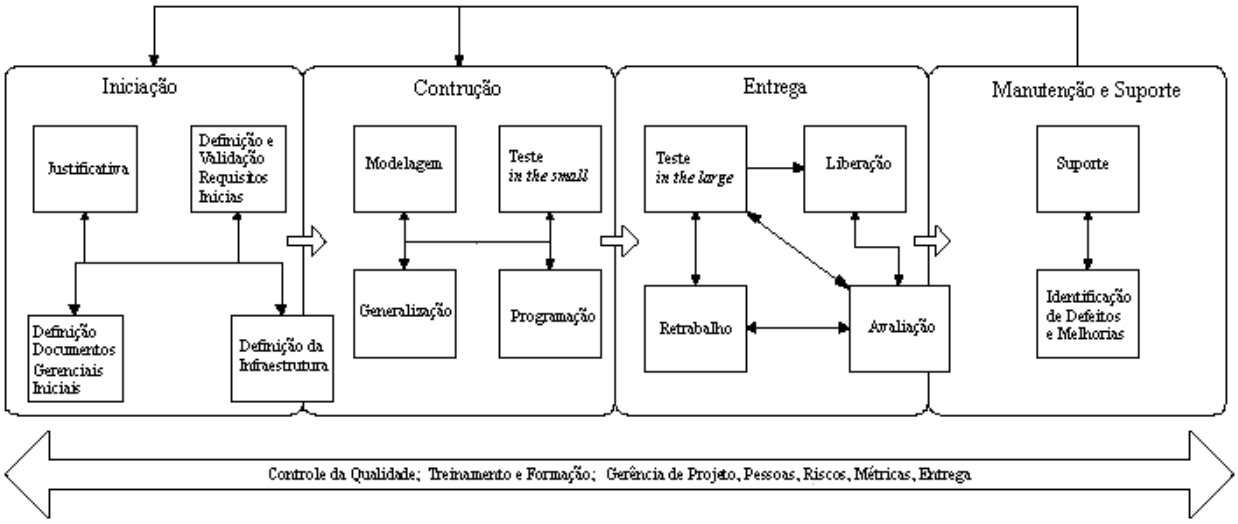

Figura 2 – O ciclo de vida do OOSP

Segundo [Ambler00], existem três níveis de padrões de processo:

- Padrões referentes às fases de um processo: descrevem as interações entre os diferentes estágios dentro de uma mesma fase de um projeto. Por exemplo, as fases de Iniciação e Construção, mostradas na Figura 2.
- Padrões referentes aos diferentes estágios dentro de uma fase: descrevem quais tarefas devem ser realizadas para um determinado estágio de uma fase. Por exemplo, as tarefas para os estágios de Modelagem e Programação, na fase de Construção.

- Padrões referentes às tarefas a serem realizadas: descrevem aspectos de um processo num nível menos abstrato, mais específico. Por exemplo, o padrão de processo *Reuse First*, que descreve como se atingir um nível de reuso significativo para uma organização; ou o padrão *Technical Review*, que descreve como organizar revisões e inspeções.

Como mostrado na Figura 2, o OOSP é composto por quatro fases e quatorze estágios divididos por estas fases, cada um descrito por um padrão. Os estágios são executados de forma interativa, dentro do escopo da fase da qual fazem parte. As fases são executadas em série. As tarefas mostradas na parte inferior da Figura 2 aplicam-se a todas as fases do desenvolvimento e são muito importantes para o sucesso do projeto. Dentre essas tarefas, estão o Controle da Qualidade, Treinamento e Formação dos profissionais, Gerência de Pessoas, Riscos, Reuso, Métricas, Entrega e Infra-estrutura. Muitas dessas tarefas aplicam-se tanto a um projeto quanto ao conjunto de projetos em execução de uma organização. Por exemplo, a Gerência de Riscos tanto deve ser realizada para cada projeto como deve analisar os riscos para o conjunto deles.

Como pontos fortes do OOSP, podemos citar a cobertura completa do ciclo de vida de um projeto, incluindo tarefas para manutenção e suporte; como no OPEN, as atividades de Gerência de Programas e Infra-estrutura comuns a vários projetos possibilitam um maior reuso e facilitam a gerência de todos os projetos em andamento; o processo inclui, ainda, várias tarefas que auxiliam o desenvolvedor a obter resultados com mais qualidade, como o estágio de Generalização, Gerência da Qualidade e dos Riscos.

Assim como o OPEN, o OOSP é resultado da experiência de vários desenvolvedores em projetos bastante distintos, pequenos, médios ou grandes e em diferentes áreas tais como telecomunicações, finanças, internet, militar, governamental. Este processo surgiu da junção de experiências no mundo real e das teorias de engenharia de software mais atuais.

2.1.3 UP

(Unified Process) [Jacobson+99]

Processo criado pela *Rational Corporation* em 1998 a partir de um outro processo, o *Objectory*, enriquecido com novas práticas como a Gestão de requisitos e a Gestão de Configurações e Mudanças. A *Rational* criou também um produto comercial baseado no UP chamado RUP (*Rational Unified Process*), que traz uma base de conhecimento para auxiliar na aplicação do processo.

O processo utiliza a UML como notação para os modelos que serão produzidos ao longo do projeto. No UP, com um ciclo de vida iterativo e incremental, o desenvolvimento é visto como uma série de iterações que abrangem todo o sistema. Cada iteração consiste de um ou mais dos seguintes componentes: levantamento dos requisitos, análise, desenho, implementação, testes e desdobramento. Os desenvolvedores não supõem que todos os requisitos serão conhecidos no início do ciclo de vida, por isso mudanças são previstas durante todas as fases. As características básicas do UP são: ser centrado na arquitetura, ser dirigido a casos de uso, ser baseado em

[1] Process patterns.

técnicas de orientação a objetos, ser configurável, ter controle da qualidade e gestão de riscos.

No UP, o ciclo de vida do desenvolvimento de sistemas pode ser dividido em fases. Uma fase é um intervalo de tempo entre dois marcos do processo onde um conjunto bem definido de objetivos é encontrado, artefatos são finalizados, e decisões são tomadas para se passar ou não para a próxima fase. As fases são:

- Concepção – especificação do escopo do projeto e do modelo de negócio;
- Elaboração – especificação do plano detalhado do sistema e da arquitetura a ser utilizada;
- Construção – construção do produto completo em uma série de iterações incrementais;
- Transição – disponibilização do software para a comunidade usuária (empacotamento, instalação, configuração, treinamento, suporte técnico, etc.).

Para cada fase do processo, existe um número de iterações. Uma iteração representa um ciclo completo de desenvolvimento, que vai desde o levantamento dos requisitos até a implementação e os testes, e tem como resultado uma liberação de executável.

A Figura 3 mostra a distribuição do esforço em cada fluxo, dentro de cada uma das fases do projeto. Esta figura se refere ao RUP, que descende do UP. Além dos fluxos do UP (Requisitos, Análise, Desenho, Implementação e Testes), o RUP apresenta outros fluxos de processo e os fluxos de suporte responsáveis pelos aspectos gerenciais e de infra-estrutura. Na parte de baixo da figura foram mostradas as várias iterações dentro de cada uma das fases.

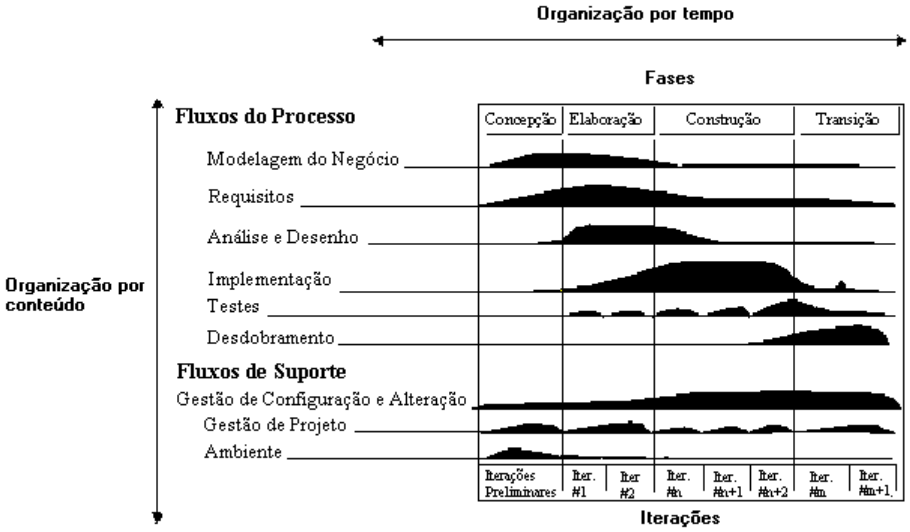

Figura 3 – O ciclo de vida do RUP

Como ponto forte do UP, pode-se citar o suporte à redução de riscos em um projeto. Este suporte baseia-se nos resultados produzidos em cada iteração, que podem ser testados e checados até mesmo pelos usuários, e também na avaliação feita ao final de cada fase para definir se o projeto segue seu curso ou algo tem que ser repensado. Estes aspectos facilitam também o trabalho dos gerentes, que podem ter, a cada iteração, uma boa noção do andamento do projeto. Outro ponto positivo é que o processo é baseado em princípios já consagrados da engenharia de software como o desenvolvimento iterativo, o fato de ser dirigido pelos requisitos do sistema e

ter o desenvolvimento baseado numa arquitetura definida para o sistema.

Mas, segundo [Ambler00], existem também algumas falhas. Ao contrário do OPEN e do OOSP, o UP não abrange todo o ciclo de vida de um projeto, mas somente o ciclo de desenvolvimento. Percebe-se a falta de uma fase depois da de Transição para o período em que o sistema já está em operação, trazendo atividades de suporte aos usuários, procedimentos para o caso de manutenções, etc. Faltam, também, atividades comuns a diferentes projetos que estejam sendo desenvolvidos, o que dificulta o reuso em larga escala na organização. [Ambler00] sugere algumas modificações no RUP para resolver estes problemas herdados do UP. As modificações são apresentadas na Figura 4.

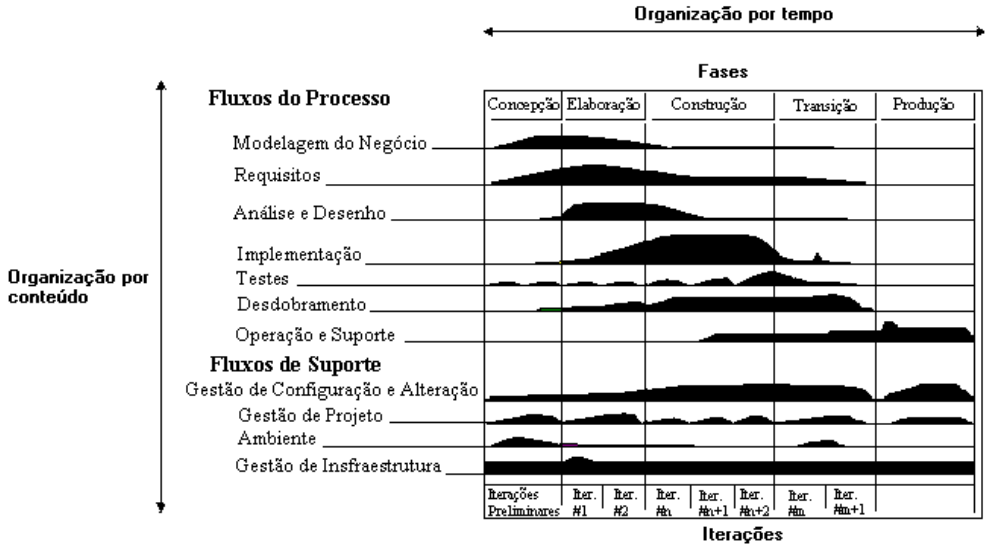

Figura 4 – O ciclo de vida para o RUP modificado

A fase de Produção incluída corresponde ao tempo de vida do sistema depois que ele já foi distribuído para seus usuários. Esta fase possui uma única iteração e seu propósito é manter o software em produção até que ele seja substituído por uma nova versão ou descontinuado.

Foram incluídos também dois novos fluxos: Operação e Suporte e Gestão de Infra-estrutura. O primeiro define como devem ser estas duas atividades – operação e suporte – ao longo das fases. Na Construção, deverão ser criados os planos e documentos para a operação e suporte e também manuais para treinamento. Na Transição, refinam-se os artefatos produzidos na Construção com base nos testes e treina-se a equipe que irá trabalhar com o suporte no software produzido. Na Produção, a equipe de operação ficará responsável por manter tudo funcionando e fazer *backups*, e a equipe de suporte trabalhará diretamente com os usuários do sistema. Quando se trata de Internet, o trabalho da equipe de operação é imprescindível, pois são sistemas que precisam estar disponíveis 7 dias por semana, 24 horas por dia. O segundo fluxo incluído diz respeito à manutenção de aspectos comuns a vários projetos, como a melhoria e suporte dos processos da organização, padrões, métricas, reuso feito de projeto para projeto, etc.

Outras modificações estenderam cada um dos fluxos para a nova fase e alguns foram estendidos para fases já existentes, como o fluxo de Testes trazido também para a fase de Concepção.

Com estas alterações, o RUP passa a suprir seus antigos pontos fracos e fica mais próximo

dos dois outros processos apresentados, o OPEN e o OOSP.

2.1.4 PSP e TSP

PSP (Personal Software Process) [Humphrey95][Humphrey97]

Conjunto de processos (PSP0, PSP0.1, PSP1, PSP1.1, PSP2, PSP2.1, PSP3) criados para melhoria da prática de processos no nível dos desenvolvedores individuais. Para se capacitar, o desenvolvedor deve realizar os projetos contidos em [Humphrey95], seguindo rigorosamente os processos. Os projetos são individuais, com duração típica de 10 horas, e trazem um conjunto de formulários, relatórios e scripts que o desenvolvedor deverá utilizar.

Após realizar todos os projetos, o desenvolvedor terá se familiarizado e se capacitado a realizar as atividades de:

- Planejamento: inclui estimativas de tamanho (linhas de código), de esforços (tempo de desenvolvimento), de cronograma (tempo físico) e de defeitos.
- Desenho: utilizando conceitos da orientação a objetos, síntese lógica e máquinas seqüenciais.
- Desenvolvimento: constituída pelo desenho detalhado, codificação, revisão do código, compilação e testes.
- *Post-mortem*: fase em que é feito um balanço final do projeto, sendo identificados pontos positivos e negativos para que se possa melhorar o processo no projeto seguinte.

Uma seqüência do PSP é apresentada em [Humphrey99], o Processo de Software para Times, TSP (*Team Software Process*). Neste processo, os desenvolvedores são organizados de forma a desempenhar um ou dois papéis gerenciais e também participar do desenvolvimento. Como no PSP, são feitos o planejamento e o controle de tamanhos, esforços, prazos e defeitos. São ainda enfatizadas algumas áreas do nível 2 do CMM[1], dentre elas: gestão dos requisitos, planejamento e controle de projetos, garantia da qualidade e gestão de configurações.

2.1.5 PRAXIS

(PRocesso para Aplicativos eXtensíveis e InterativoS) [Paula01]

O Praxis é um processo completo de desenvolvimento de software adequado para projetos com duração de seis meses a um ano, em um contexto educacional ou de treinamento.

Este processo traz elementos do PSP, UP, TSP e PROSE (Processo Orientado a Objetos para Software Extensível) [Paula+98a][Paula+98b][Paula+98c][Paula+98d], utiliza a UML como notação, e inspira-se nas práticas chaves dos níveis 2 e 3 do SW-CMM[2]. Os padrões incluídos procuram ser conformes com os padrões correspondentes do IEEE [IEEE94].

---

[1] *Capability Maturity Model.* Modelo de maturidade que visa caracterizar o nível de capacitação de uma organização no contexto da engenharia de software. Foi proposto pelo *Software Engineering Institute* da universidade *Carnegie-Mellon* [Paulk+95].

[2] CMM voltado para organizações produtoras de software.

Assim como no UP, o ciclo do desenvolvimento de uma aplicação é dividido em fases (divisões de caráter gerencial), sendo elas: concepção, elaboração, construção e transição. Cada fase possui uma ou mais iterações. Uma iteração representa um ciclo completo de desenvolvimento, passando por todos os fluxos (divisões de caráter técnico). Os fluxos podem ser divididos em dois grupos: fluxos técnicos – requisitos, análise, desenho, implementação, testes – e fluxos gerenciais – gestão de projetos e gestão da qualidade. Ao final de uma iteração, têm-se resultados que serão avaliados segundo os critérios de aprovação. Em uma iteração, existem os artefatos de entrada (insumos), os artefatos que serão produzidos por ela, as pré-condições que devem ser atendidas para que a iteração se inicie e os critérios pelos quais se poderá avaliar se a iteração pode ser finalizada.

O material do PRAXIS inclui modelos para os documentos a serem produzidos durante as iterações das diferentes fases como a especificação de requisitos, o plano da qualidade, o plano de desenvolvimento, etc. Inclui também modelos de relatórios a serem preenchidos nas revisões que devem ser feitas ao final de uma iteração.

Para cada fase, o PRAXIS define as iterações mais importantes. Dessa forma temos:

- Fase da Concepção
  - Iteração de Ativação: definição do escopo do produto, levantamento preliminar dos requisitos e estimativas de custo e prazo da fase de Elaboração. Nessa iteração será produzida uma Proposta de Especificação de Software.
- Fase da Elaboração
  - Iteração de Levantamento dos Requisitos: visa à captura das necessidades dos usuários em relação ao produto, expressas na linguagem destes usuários. Nessa iteração serão produzidas partes do documento de Especificação de Requisitos do Software.
  - Iteração de Análise dos Requisitos: confecciona um modelo conceitual do produto, que é usado para validar os requisitos levantados e para planejar o desenvolvimento posterior. Nessa fase será completado o modelo de análise e dessa forma a Especificação de Requisitos. Serão ainda produzidos os planos de Desenvolvimento e da Qualidade.
- Fase da Construção
  - Desenho Inicial: definição interna e externa dos componentes do produto, em nível de se decidir as principais questões de arquitetura e tecnologia, permitindo também o planejamento detalhado das atividades de implementação. Serão produzidos o modelo de desenho do software (em alto nível), parte da descrição do desenho e a descrição dos testes de aceitação.
  - Liberação: implementação de um subconjunto de funções do produto que será avaliado pelos usuários. Serão produzidos relatórios de testes de unidade e integração da liberação, os códigos fonte e executáveis do software.
  - Testes Alfa: realização dos testes de aceitação no ambiente dos desenvolvedores, juntamente com a elaboração da documentação de usuário e possíveis planos de Transição. Serão produzidos os relatórios dos testes alfa e o manual do usuário do software.
- Fase da Transição

- Testes Beta: testes de aceitação feitos, agora, no ambiente dos usuários. Serão produzidos os relatórios dos testes beta.
- Operação Piloto: operação assistida do produto, já instalado no ambiente do cliente. Os problemas encontrados nessa fase são resolvidos através do processo de manutenção. Será produzido um relatório final do projeto.

Como citado anteriormente, os resultados produzidos e insumos consumidos nos passos do PRAXIS são chamados de artefatos. Os artefatos podem ser documentos ou modelos e podem ser permanentes ou transitórios. Os artefatos permanentes são atualizados a cada iteração, de acordo com procedimentos de gestão de configurações.

O PRAXIS possui artefatos permanentes oficiais incluindo documentos, modelos e relatórios (tipo especial de documento). O formato desses documentos é conforme com os padrões do IEEE. Para facilitar sua elaboração, o processo possui gabaritos com algumas partes previamente preenchidas (aquelas que não variam muito entre projetos) e somente as exceções precisarão ser documentadas.

Para que uma iteração em uma das fases do projeto seja dada por finalizada e a próxima iteração possa ser iniciada, devem ser realizados os procedimentos de controle. Como procedimentos de controle praticados no PRAXIS, pode-se citar:

- revisões técnicas: principal meio de controle da qualidade quanto aos aspectos técnicos. Os resultados produzidos são submetidos a um grupo de revisores, que podem aprovar o material, com modificações ou não, ou ainda reprová-lo.
- inspeções: tipo de revisão técnica, mais rigorosa, usada na revisão de desenho detalhado e código.
- revisões gerenciais: o gerente de projeto determina se a atividade pode ser finalizada, com base no depoimento dos membros de sua equipe envolvidos. Se for mesmo finalizada, o gerente conduz um balanço das atividades da iteração e toma as medidas necessárias para que a próxima se inicie. Se não puder ser finalizada, o gerente irá solicitar que a atividade seja refeita.
- auditorias da qualidade: realizadas por um grupo independente da qualidade, que checa a conformidade das atividades realizadas com os padrões determinados pelo processo.

Algumas iterações requerem a aprovação por parte do cliente ou dos usuários chave. O primeiro caso acontece quando estão envolvidas decisões de se continuar ou não o projeto (fim da Concepção e Elaboração) ou na aceitação final do produto (fim da Construção e Transição). O segundo caso acontece quando é necessário verificar se o produto atende às necessidades dos usuários, em algum dos estágios de sua construção.

2.2 Desenvolvimento de Sistemas Hipermídia

2.2.1 RMM

(Relationship Management Methodology) [Isakowitz+95]

RMM é uma metodologia proposta para projeto e construção de aplicações hipermídia. O nome *relationship management* enfatiza a visão dos autores de que a hipermídia é um veículo para gerenciar relacionamentos entre os objetos de informação.

A classe de aplicações onde o RMM pode ser melhor aplicado apresenta uma estrutura regular do domínio de interesse, com classes de objetos, relacionamentos definidos entre essas classes e objetos que instanciam essas classes. Muitas aplicações hipermídia se encaixam nesse perfil.

A seqüência de passos a serem seguidos no projeto de uma aplicação seria:

- Representar o domínio da aplicação através de um diagrama E-R, levantando as entidades e os relacionamentos que são importantes para a mesma. O objetivo deve ser sempre de explicitar os *links* entre objetos, pois tais *links* serão as principais vias de acesso do usuário aos diferentes objetos.
- Determinar como a informação contida nas entidades definidas previamente será exibida para o usuário e como estes irão acessá-la. Isto é feito através de um diagrama chamado de *slice diagram*, onde cada entidade é subdividida em partes menores e essas partes são organizadas em termos de uma rede hipertexto (que mostra, basicamente, como essas partes serão exibidas e como se interligam).
- Projeto navegacional: projetar as ligações que irão possibilitar a navegação hipertexto. Cada relacionamento do diagrama E-R é analisado. Se for importante que um relacionamento seja disponibilizado para acesso do usuário, ele é substituído por uma ou mais estruturas de acesso (link unidirecional, link bidirecional, agrupamento, índice condicional, roteiro guiado). O resultado é um diagrama chamado de *RMDM diagram*.
- Traduzir cada elemento do diagrama RMDM em um objeto da plataforma a ser utilizada através de regras de conversão.
- Projeto de interface de usuário: para cada objeto do diagrama RMDM criar um *layout* de tela onde os dados serão apresentados.
- Projetar como cada elemento de navegação será implementado.
- Construir e testar.

2.2.2 OOHDM

(The Object-Oriented Hypermedia Design Method) [Rossi96][Schwabe+99]

Método voltado para o desenvolvimento de aplicações hipermídia, o OOHDM tem tido grande aceitação na comunidade acadêmica internacional.

O OOHDM propõe quatro atividades durante a construção de uma aplicação hipermídia:

- Modelagem conceitual: atividade responsável pela análise do domínio da aplicação.

Seu resultado será um esquema conceitual contendo classes, relacionamentos e subsistemas. Antes de elaborar o esquema conceitual, pode ser necessária a definição de cenários, os quais explicitam como um usuário executará uma tarefa com o auxílio da aplicação. A notação utilizada nessa fase se baseia na UML. O esquema conceitual será semelhante ao diagrama de classes da UML. Para definir os cenários podem ser utilizados diagramas de caso de uso.

- Modelagem navegacional: atividade que define as informações a serem apresentadas aos usuários e a possível navegação entre elas. Nela, serão especificados quais objetos navegacionais (nós, elos, classes em contexto, estrutura de dados, etc.) serão vistos pelo usuário e quais são os contextos navegacionais. Serão produzidos um esquema navegacional (definição dos objetos navegacionais), um esquema de contextos navegacionais e também cartões especificando cada um dos objetos criados. O esquema navegacional deriva-se da modelagem conceitual, sendo uma visão sobre essa modelagem. A partir desse esquema, são definidos os contextos navegacionais, as estruturas de acesso a esses contextos e as ligações entre tais contextos.
- Projeto da interface abstrata: atividade que define como serão os objetos de interface, suas propriedades e transformações. OOHDM utiliza *Abstract Data View* (ADV) para especificar o modelo de interface abstrata.
- Implementação: atividade responsável por traduzir o projeto navegacional e de interface para um determinado ambiente de implementação.

As três primeiras atividades são desenvolvidas iterativamente, enquanto a atividade de implementação geralmente é desenvolvida após o término dessas.

Segundo [Schwabe+98], a maior contribuição da OOHDM está na modelagem navegacional. No OOHDM, uma aplicação é vista como uma visão navegacional do modelo conceitual. Esta visão é construída durante a modelagem navegacional levando-se em conta os tipos de usuários aos quais a aplicação se destina e o conjunto de tarefas que deverão desempenhar ao utilizá-lo.

Após tentarmos simplificar a modelagem navegacional e representá-la tendo a UML como notação, percebemos que a melhor opção seria mesmo utilizá-la na íntegra, como apresentada na OOHDM, de forma ortogonal ao PRAXIS. Assim, para aquelas aplicações onde se faz necessária uma modelagem mais detalhada de como o usuário poderá navegar e ter acesso às informações, a modelagem conceitual do PRAXIS pode ser complementada com a modelagem navegacional da OOHDM.

Capítulo 3

Modelagem de aplicações Web

3.1 Introdução

Segundo [Conallen99a], uma aplicação web pode ser definida como um sistema de software cuja interface com os usuários é feita através de um sistema Web e que permite que as ações desses usuários afetem o estado da lógica de negócio[1]. Um sistema Web é composto, basicamente, por um servidor Web, um protocolo de comunicação – sendo o HTTP o mais utilizado – e um navegador. As referidas ações dos usuários que alteram o estado do negócio[2] são a navegação e a entrada de dados, feitas através do navegador.

A arquitetura de uma aplicação web não difere muito da arquitetura de um sítio dinâmico. As diferenças estão na forma como são utilizados e em quais casos um ou outro é aplicável. Aplicações web implementam lógica de negócio e ao serem utilizadas pelos usuários, o estado do negócio é alterado. Esse aspecto é muito importante, pois define qual deve ser o foco dos esforços para se modelar o sistema [Conallen99a]. Em um sítio dinâmico, os esforços de modelagem resumem-se praticamente a aspectos de apresentação das páginas e a navegação entre elas. Numa aplicação web, os esforços de modelagem devem concentrar-se na lógica de negócio e no estado do negócio. A apresentação é também importante, mas deve ser considerada e modelada separadamente, sem fazer parte do modelo da lógica de negócio. Esta separação justifica-se, pois os aspectos e recursos empregados na apresentação tendem a ser mais artísticos e são independentes da implementação das regras de negócio.

O RMM e o OOHDM apresentados em Trabalhos Relacionados, trazem uma metodologia e notações próprias que se adequam à modelagem de sítios dinâmicos, pois tratam mais a fundo aspectos de apresentação. Como as aplicações web são centradas na lógica de negócio, elas

[1] Entenda-se por "lógica de negócio" o conjunto de regras que define a forma como as ações devem ser feitas, como devem ser interpretadas pela aplicação e como os dados devem ser tratados para produzir as saídas esperadas. Por exemplo, uma validação de que a data de pagamento deve ser anterior ou igual à data de vencimento faz parte da lógica de negócio de uma aplicação.

[2] Entenda-se por "estado do negócio" o conjunto de valores que possam descrever o ambiente da aplicação num dado momento. De acordo com o estado do negócio (situação atual) e com as regras da lógica de negócio, a aplicação decide qual o próximo passo a ser feito. Por exemplo, considere a regra de negócio descrita em 1, se um usuário entrar com uma data de pagamento maior que a de vencimento, ele afetará o estado do negócio. A aplicação passará de uma situação válida para uma inválida e deverá avisar o usuário que a data não pode ser aceita.

incluem vários mecanismos para implementar esta lógica que não são cobertos pela notação apresentada nestes trabalhos. Dentre esses mecanismos, podemos citar *scripts*, *applets*, controles *ActiveX*, utilização de várias instâncias do navegador, utilização de *frames*, etc. Estes e vários outros mecanismos, próprios de uma aplicação web, representam uma parte da lógica de negócio e precisam ser incluídos e integrados ao restante dos modelos do sistema.

Na maioria das vezes, grande parte da lógica de negócio de uma aplicação web é executada fora do sistema web, em uma das camadas do lado do servidor. A escolha de uma notação a ser utilizada na modelagem deve então ser baseada nas necessidades desse lado da aplicação. Assim, podemos justificar a escolha da UML[1] acrescida de estereótipos que possibilitem a modelagem dos mecanismos específicos de uma aplicação web. A UML vem sendo cada vez mais utilizada na modelagem de sistemas tradicionais e tornou-se quase que obrigatória para que as diversas organizações tenham uma linguagem comum de comunicação de suas experiências. Por outro lado, a UML é a notação utilizada pelo PRAXIS e a notação escolhida para um processo derivado, no caso o WebPraxis, deve ser, de preferência, compatível com a UML.

3.2 Aspectos Importantes

As aplicações web estão se tornando cada vez mais complexas e sendo utilizadas cada vez mais em áreas críticas das organizações. Uma forma de gerenciar este aumento de complexidade em sistemas de software é criar abstrações desses sistemas e modelá-los. Pode-se criar diferentes modelos para um sistema, para representar diferentes pontos de vista e níveis de abstração. Os modelos ajudarão a entender o sistema, simplificando alguns de seus detalhes [Conallen99a].

As aplicações web, assim como outros sistemas, podem ser representadas por um conjunto de modelos: modelo de casos de uso, modelo de implementação, modelo de desdobramento, modelo de segurança e ainda um modelo adicional muitas vezes chamado de mapa do sítio, que traz as páginas e mostra as possibilidades de navegação entre elas. A confecção destes modelos para uma aplicação Web não difere muito de sua confecção para sistemas tradicionais. As maiores dificuldades serão sentidas naqueles pontos onde são modeladas as páginas de uma aplicação web e a lógica envolvendo sua execução. Normalmente, isto acontece nos modelos de análise e desenho.

Ao se pensar em modelar um sistema, é importante definir o que será modelado, o nível de abstração adotado, para que o resultado realmente agregue algum valor ao seu processo de desenvolvimento. Com relação aos aspectos específicos de uma aplicação web, o importante é modelar as páginas, hiperligações e o conteúdo dinâmico das páginas no nível do cliente e do servidor, deixando-se para traz detalhes internos do servidor web ou do navegador.

Pode-se entender melhor o que modelar e porque modelar estes elementos ao se analisar a arquitetura de uma aplicação web. O básico desta arquitetura inclui navegadores, uma rede e um servidor web. Pode-se comparar essa arquitetura a uma típica cliente/servidor, onde, do lado cliente, estão os navegadores e do lado do servidor pode-se ter um único servidor web ou ainda um servidor web com um servidor separado para os dados, etc. Os navegadores fazem requisições de páginas ao servidor. Este, por sua vez, faz o processamento necessário e devolve a página

[1] *Unified Modeling Language*, [Booch+99].

gerada por este processamento como uma mistura de conteúdo e instruções de formatação expressas em HTML. Algumas páginas podem incluir também *scripts* que serão interpretados pelo navegador. Estes scripts definem comportamentos adicionais para a página e interagem com o navegador, o conteúdo da página e outros controles que podem estar presentes na página, como controles *ActiveX* e *plug-ins*. Os usuários interagem com o sistema através do conteúdo das páginas, algumas vezes, informando dados que serão enviados para o servidor para processamento, ou ainda navegando pelas diferentes páginas do sistema.

As páginas web podem ser classificadas de acordo com o processamento a ser feito pelo servidor para produzi-las. Assim, podemos dividi-las em páginas de script, páginas compiladas e páginas híbridas (uma mistura dos dois outros tipos) [Conallen99a]. As páginas de *script* são armazenadas no servidor como um arquivo contendo uma mistura de HTML com alguma linguagem de *script*. Quando uma destas páginas é requisitada, o servidor delega o processamento da página para um mecanismo capaz de interpretar o *script* e retorna um arquivo em HTML para o requisitante. Exemplos de páginas de script são páginas ASP (*Active Server Pages*) e *Cold Fusion*.

Para as páginas compiladas, o processamento do servidor já é outro. Quando uma delas é requisitada, o servidor carrega e executa um componente já compilado e armazenado. Este componente utiliza os valores e parâmetros passados com a requisição, e, na maioria das vezes, acessa também recursos do lado do servidor, para produzir a página HTML como resposta à requisição. Tipicamente, páginas compiladas englobam um número maior de funcionalidades do que páginas de script. Estas diferentes funcionalidades podem ser obtidas passando-se diferentes parâmetros na requisição da página. Exemplos deste segundo tipo são *Microsoft ISAPI* e *Netscape NSAPI*.

As páginas híbridas são páginas de script que são compiladas quando é feita uma requisição e esta versão compilada é utilizada pelas requisições subseqüentes. Quando acontece alguma alteração no conteúdo da página original, ela é compilada novamente. Exemplos desse tipo de página são páginas JSP (*Java Server Pages*).

3.3 Notação Adotada

A notação adotada para o WebPraxis é a WAE (Web Application Extension for UML), que estende a notação UML trazendo novos estereótipos com semântica e restrições adicionais capazes de modelar os aspectos levantados anteriormente.

Como dito na seção 3.1, os aspectos específicos de uma aplicação web que devem ser modelados são as páginas, hiperligações e o conteúdo dinâmico das páginas no nível do cliente e do servidor. Nessa seção, pretende-se mostrar como estes artefatos foram mapeados para elementos de modelagem pela WAE. Não serão apresentados todos os novos estereótipos, mas alguns básicos que mostram as decisões tomadas por Jim Conallen ao criar esta extensão. Para uma lista completa dos estereótipos introduzidos pela WAE e maiores detalhes sobre os estereótipos, valores rotulados e restrições, direcionamos o leitor para [Conallen99a] e [Conallen99b].

Páginas web têm um mapeamento um-para-um para componentes na UML [Conallen99a]. Para uma aplicação web, o diagrama de componentes da UML conteria todas as páginas do

sistema e os relacionamentos entre elas (hiperligações), se assemelhando a um mapa do sítio.

Em um diagrama de componentes são representados somente aspectos físicos, ou seja, como os elementos do sistema estão encapsulados fisicamente. Sendo assim, tal diagrama não é apropriado para modelar o processamento e as colaborações que ocorrem nas páginas. Por isso, é necessário criar um modelo para a visão lógica do sistema. Nesta visão lógica, as páginas são mapeadas para classes. Assim, os scripts contidos pela página seriam mapeados para operações da classe e as variáveis contidas nesses scripts, que tivessem escopo de página, seriam mapeadas para atributos da classe.

Um problema a ser resolvido é que uma página pode representar um conjunto de funções e colaborações válidas somente do lado do servidor e um outro conjunto completamente diferente, válido somente no cliente. Para resolvê-lo, devemos levar em consideração que, em termos de lógica, o comportamento de uma página no servidor é diferente de seu comportamento no cliente. Quando está sendo executada no servidor, a página se relaciona com recursos do lado do servidor como componentes, bancos de dados e sistema arquivos. Quando apresentada no cliente, a página se relaciona com o navegador, com *applets Java*, controles *ActiveX* e *plug-ins* especificados na página.

A solução é tratar estes dois comportamentos separadamente. As características de uma página no lado do servidor são mapeadas para uma classe e as do lado do cliente são mapeadas para outra classe. Para distinguir as duas classes, foram incluídos na WAE os estereótipos «server page» e «client page». As duas abstrações de uma mesma página web estão vinculadas entre si através de um relacionamento direcional, partindo da servidora para a cliente. Para este relacionamento, também foi criado um estereótipo: «build». Assim, toda página dinâmica (cujo conteúdo é determinado em tempo de execução) é construída por uma página servidora. Uma página servidora pode construir uma ou mais páginas clientes, dependendo dos parâmetros passados na requisição.

Uma das principais vantagens de se separar aspectos do servidor de aspectos do cliente em duas classes distintas está na representação dos relacionamentos dessas classes com outros elementos do sistema. As páginas são inseridas no modelo e suas colaborações com os outros elementos do sistema são expressas de forma semelhante a qualquer colaboração entre elementos que não são específicos da web.

Um relacionamento comum entre duas páginas é a hiperligação, que representa a possibilidade de se navegar de uma página a outra dentro do sistema. Estes relacionamentos são mapeados para associações estereotipadas por «link» no modelo. Uma associação do tipo «link» deve partir de uma página cliente para uma página cliente ou servidora. Isto acontece pois a associação «link» representa uma requisição de uma página e, sendo assim, podemos usar qualquer uma das duas abstrações de página como resposta a essa requisição. Para definir os parâmetros passados em uma associação «link», foi criado um valor rotulado "*Parameters*", que é uma lista de parâmetros com seus nomes e valores (opcional). Estes parâmetros são utilizados pelo servidor para processar a requisição da página.

A principal forma de entrada de dados numa página web são os formulários (especificado por <form> em HTML). A WAE traz um estereótipo específico para os formulários. Estes são mapeados para classes estereotipadas por «Form». Uma classe «Form» não possui operações e as entradas de dados do formulário são atributos estereotipados com nomes semelhantes aos usados

em HTML. Uma classe «Form» pode se relacionar com A*pplets* e controles *ActiveX* e tem um relacionamento com a página servidora responsável por processar o formulário. Para o relacionamento do formulário com a página que o processa, foi criado o estereótipo «submit». O relacionamento do formulário com a página que o contém é sempre expresso como uma composição, pois um formulário sempre faz parte de alguma página, ele não existe de forma independente.

Outros dois estereótipos de classe incluídos são «frameset» e «target» e, ainda, um estereótipo de associação «targeted link» usados para modelar a utilização de frames numa aplicação web. Uma classe «frameset» representa um contêiner que corresponde diretamente ao rótulo <frameset> do HTML e contém páginas clientes e frames (classes «target»). Uma classe «target» é um frame ou uma instância de navegador referenciada por páginas clientes. Uma associação «targeted link» é uma hiperligação para outras páginas que serão exibidas em uma classe «target» específica. Para expressar em qual «target» uma página será exibida, foi criado um valor rotulado na associação «targeted link»: "Target". No relacionamento de agregação entre um «frameset» e um «target» ou página cliente, foram criados os valores rotulados "*row*" e "*col*" que especificam a localização destes dentro do «frameset». Quando se deseja expressar que uma classe «target» corresponde a uma instância diferente do navegador, basta não criar o relacionamento entre ela e a classe «frameset».

A Figura 5 mostra um exemplo simples de modelagem onde foram utilizados alguns dos estereótipos introduzidos pela WAE. O usuário acessa a página inicial e informa os dados de um de seus fornecedores, o nome e/ou CNPJ. O formulário onde a pessoa irá entrar com esses dados é modelado com o estereótipo «Form», com atributos também estereotipados e que tem um relacionamento «submit» com a página servidora que irá processar esses dados.

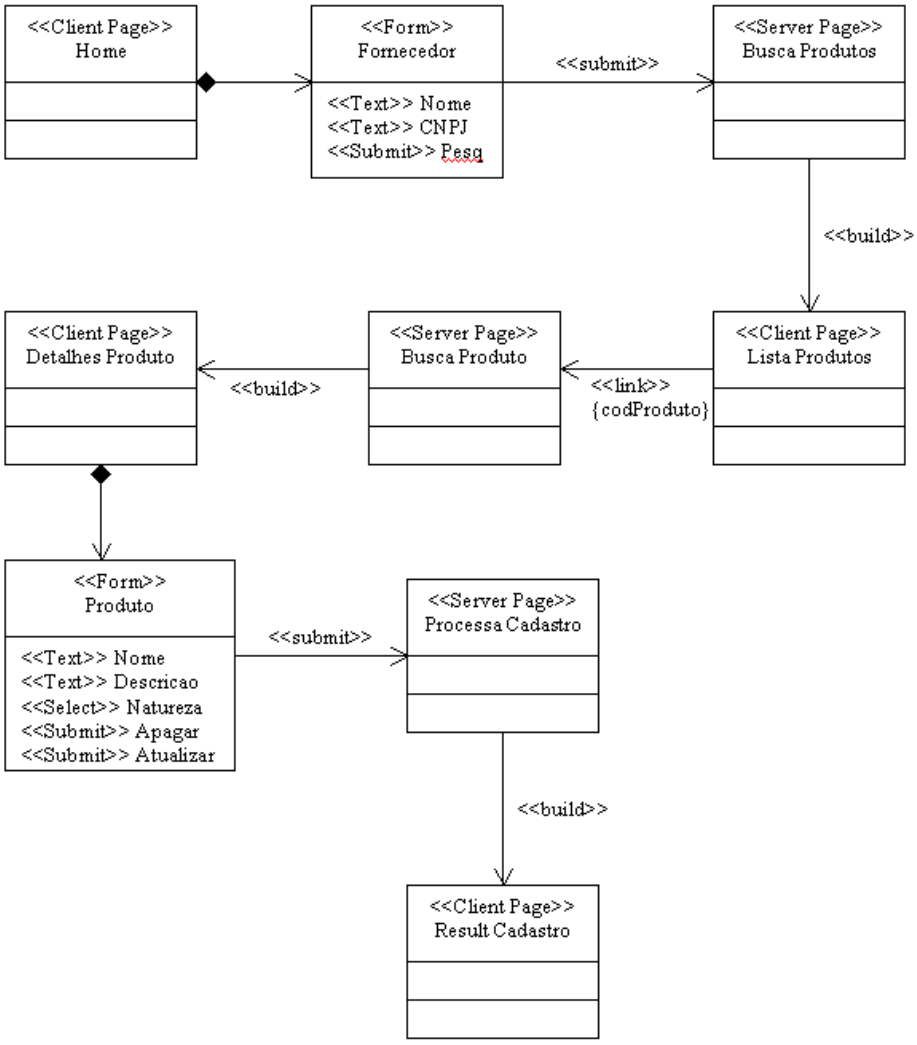

Figura 5 – Exemplo de modelagem utilizando estereótipos introduzidos pela WAE

Esta página, a "Busca Produtos", irá buscar todos os produtos cadastrados para o fornecedor informado e construirá uma página cliente (relacionada à servidora através de uma associação «build») com a lista dos produtos encontrados. O usuário poderá escolher um dos produtos para visualizar ou editar seus dados, ou mesmo apagar este produto. Ao escolher um dos produtos será apresentada ao usuário uma página com os dados desse produto. O processamento feito para exibir esta página é modelado com um relacionamento «link» entre "Lista Produtos" e "Busca Produto", passando-se o parâmetro codProduto, que identifica o produto escolhido. A página servidora "Busca Produto" obtém os dados para aquele produto e constrói a página cliente a ser exibida para o usuário. Esta, por sua vez, possui um formulário onde o usuário poderá alterar os dados do produto ou apagar o produto. A página servidora "Processa Cadastro" irá processar as ações feitas no formulário e construir para o usuário uma página de resposta, confirmando ou não estas ações.

O objetivo desse exemplo simples é mostrar a utilização dos novos estereótipos e por isso os atributos e operações das classes, excetuando-se os formulários, foram omitidos.

Capítulo 4

A Definição de um Processo

4.1 Introdução

Ao se pensar em definir um novo processo de desenvolvimento de software para uma organização ou melhorar um já existente, alguns fatores devem ser levados em consideração [Ambler00]. Este novo processo, ou a nova versão de um processo, deverá interagir com outros processos já existentes na organização e se adaptar a eles. Sofrerá, também, a influência de outros elementos como a cultura da organização, a arquitetura e ferramentas utilizadas, a legislação que envolve o negócio da organização, os processos de outras organizações com as quais se tem interação, dentre outros (Figura 6).

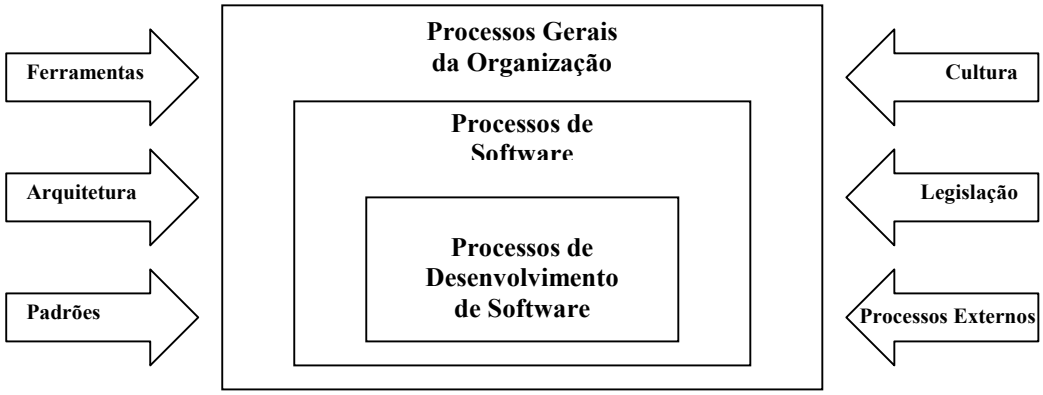

Figura 6 – Fatores que influenciam um processo de desenvolvimento de software

Um processo bem definido não é necessariamente aquele que inclui as mais novas tecnologias, e sim, um que se enquadre na estrutura e no negócio da organização. Se o novo processo não se adequar a algum dos fatores mostrados na Figura 6, ele não será realizável na prática.

Como tarefa inicial para a definição de um processo, deve-se descobrir seus requisitos, ou seja, o que precisa estar presente neste processo. A definição dos requisitos do processo pode se

espelhar, por exemplo, nos níveis do CMM. O CMM define para cada nível, as práticas chaves que devem estar presentes no processo da organização[1].

A seção seguinte traz uma abordagem mais estruturada para se conseguir definir um processo que atenda aos objetivos de uma organização.

4.2 Metodologia para a Definição de um Processo

Humphrey propõe em [Humphrey95] a metodologia para a definição de processos que foi adotada para o presente trabalho. Segundo esta metodologia, os passos que devem ser seguidos para se obter um processo que atenda às necessidades de uma organização são:

- Determinar as necessidades e prioridades do novo processo;
- Definir os objetivos e critérios da qualidade;
- Caracterizar o processo atual;
- Caracterizar o processo desejado;
- Estabelecer uma estratégia de desenvolvimento do processo;
- Definir um processo inicial;
- Validar o processo inicial.

Estes passos não precisam ser seguidos exatamente nesta ordem, mas todos devem ser realizados. A explicação de cada um deles é dada a seguir.

4.2.1 Determinar as necessidades e prioridades do novo processo

Nesta fase, devem ser estabelecidas as atividades que precisam estar presentes no processo e qual é sua ordem de prioridade, levando-se em consideração as características dos produtos a serem produzidos.

Humphrey sugere o uso do QFD (*Quality Function Deployment Method*) para relacionarmos as características do processo com as necessidades dos futuros usuários dos produtos produzidos por ele. Para isso, é necessário:

- Determinar o tipo de produtos que o processo irá produzir;
- Identificar os principais atributos desses produtos;
- Determinar a prioridade relativa desses atributos;
- Determinar as atividades necessárias no processo para produzir tais atributos;
- Classificar a ligação entre as atividades do processo e os atributos do produto em forte, média, fraca;
- Classificar as atividades do processo em prioridade alta, média, baixa.

Ao final dessas atividades, será possível determinar quais melhorias no processo agregam maior valor ao desenvolvimento, bastando comparar os valores numéricos obtidos.

Para o WebPraxis, este levantamento foi realizado e é apresentado na seção 5.1.1, deste

[1] Uma tabela completa com os 5 níveis do CMM e as respectivas práticas, pode ser encontrada em [Ambler00].

documento.

4.2.2 Definir os objetivos e critérios da qualidade

Na fase anterior, foram considerados os atributos dos produtos a serem produzidos. Agora é necessário relacionar as atividades do processo com os objetivos estabelecidos ao se pensar em criar ou melhorar o processo. Exemplos desses objetivos podem ser melhorar a previsão de custos e prazos de um projeto, ou incluir planejamento para a fase de Operação, etc.

Mais uma vez, utiliza-se a técnica do QFD, e os passos já apresentados na seção anterior, só que, agora, levando-se em consideração os objetivos do processo e não os atributos do produto final.

Ao final deste processo, é possível identificar quais melhorias no processo são mais importantes por estarem relacionadas diretamente aos objetivos de maior prioridade.

Os resultados da fase descrita na seção 4.2.1 e os desta fase podem ser combinados de diferentes formas. Pode-se considerar que os atributos do produto final e os objetivos do processo têm prioridades iguais, ou que um deles tem uma maior prioridade. De acordo com os pesos que se deseja dar a um e outro, os resultados são somados e obtêm-se, então, as melhorias que devem ser feitas no processo e em que ordem.

É importante também, nesta fase, estabelecer critérios que permitam verificar se um determinado objetivo foi atingido ou não ao final da definição do processo. Uma forma de se fazer isto, é criar *checklist* com os objetivos e, ao aplicar o processo criado, responder com um simples sim/não o que foi alcançado ou não. Mas, o ideal mesmo é estabelecer critérios numéricos, que possam ser mensurados de forma mais precisa.

Para o WebPraxis, as atividades desta fase foram realizadas e o resultado é apresentado na seção 5.1.1, deste documento.

4.2.3 Caracterizar o processo atual

Para se conseguir atingir os objetivos definidos para o processo, é necessário conhecer a situação atual da organização. É preciso saber em que ponto esta se encontra e quão distante está de seus objetivos.

Deve-se identificar as fases do processo atual, os critérios que determinam o fim de uma fase e o início de outra, qual o nível de planejamento possível de ser feito com processo atual, etc.

No caso deste trabalho, o PRAXIS foi considerado o processo atual e encontra-se caracterizado na seção 2.1.5 deste documento.

4.2.4 Caracterizar o processo desejado

Para alcançar todos os objetivos definidos para o processo e atender às necessidades dos usuários de seus produtos finais, deve-se definir as atividades, passos e fases do novo processo voltadas para esse fim.

O processo desejado inclui todas as modificações a serem feitas no processo atual que irão possibilitar alcançar os objetivos estabelecidos. Ainda que nem todas as mudanças sejam feitas de uma só vez, elas devem ser definidas aqui, para se ter uma idéia de onde se pretende chegar.

A maior parte deste trabalho se refere a esta fase. O processo desejado é o próprio WebPraxis e inclui todos os aspectos não cobertos pelo PRAXIS e que foram julgados necessários para o desenvolvimento de uma aplicação Web. A caracterização do WebPraxis se encontra no Capítulo 5 deste documento.

4.2.5 Estabelecer uma estratégia de desenvolvimento do processo

Nesta etapa, é definido um "mapa" que parte do processo atual e chega no processo desejado. Muitas vezes, o processo ideal distancia-se enormemente do processo real, o atual. Depois de definido como seria esse processo ideal, chega-se a conclusão que seria impossível simplesmente mudar de uma só vez tudo o que é necessário. É preciso, então, levando-se em consideração os recursos disponíveis, tanto de pessoas quanto de tempo e dinheiro, determinar como as mudanças serão feitas, em que ordem. Para isso, deve-se determinar a prioridade das mudanças. É preciso identificar quais mudanças são realizáveis no momento ou, ainda, quando será possível implementá-las. Com essa classificação das mudanças, serão definidos o processo inicial e os próximos passos a serem seguidos até atingir-se o processo ideal. Deve-se determinar com qual freqüência novas modificações serão incluídas, estimar prazos e possíveis custos. É necessário, também, estimar os principais impactos que as mudanças trarão para o processo atual.

No caso deste trabalho, a principal limitação é no tocante ao tempo. Tendo esse fator em mente, poder-se-ia determinar o que seria introduzido e modificado no PRAXIS e o que seria deixado para trabalhos futuros. Como dito na seção anterior, todos aqueles aspectos que foram julgados importantes para o desenvolvimento de uma aplicação web foram incluídos no processo. Isso pode ser feito por se tratar de um trabalho acadêmico. Numa organização real, é muito importante que as mudanças sejam introduzidas de forma gradual.

4.2.6 Definir um processo inicial

O processo inicial é aquele determinado como o primeiro passo na direção do processo ideal. Ele abrange as modificações classificadas como mais importantes, resultantes da primeira e segunda fases, e que são possíveis de serem implementadas no momento.

Como neste trabalho não foi preciso modificar o PRAXIS e sim especializá-lo, o processo final foi considerado igual ao inicial. Ao empregar esse processo em um projeto real,

provavelmente serão identificadas novas melhorias.

4.2.7 Validar o processo inicial

A definição de um processo só acaba quando ele é colocado em prática para ser testado. A validação do processo será feita por seus usuários. O ideal é aplicar o processo criado em um projeto piloto, onde possam ser gastos recursos extras para se experimentar novas tecnologias. A aplicação do processo em um projeto real poderá apontar aspectos que podem ser ainda melhorados e trazer também sugestões.

O estudo de caso realizado para testar o WebPraxis é apresentado no Capítulo 6. É importante deixar claro que o estudo de caso deve ser visto como uma verificação de que o processo é factível na prática, pois foi realizado por quem definiu o processo. Uma validação ainda precisa ser feita junto a outros usuários, em um projeto real.

4.3 Melhoria de Processos de Software

Ainda que seja utilizada uma metodologia para se definir um processo, obter um resultado final que cumpra seus objetivos e que realmente melhore a produção de software é muito difícil. A definição de processos é algo que se aprende e que pode ser melhorado com a prática. É necessário definir um processo e aplicá-lo na prática. Com o resultado desta experiência, é possível ir melhorando e refinando o processo, até chegar num nível satisfatório para a organização. Para evitar muitas tentativas frustradas e gastos enormes, é importante, também, aprender com os erros de outros. Scott Ambler reuniu em [Ambler00] um conjunto de dicas para aqueles que pretendem criar um novo processo ou melhorar um já existente, que podem contribuir para se alcançar o sucesso mais rapidamente. São elas:

- **Comece devagar**: mudar a forma como as pessoas trabalham é difícil, toma tempo e faz com que a produtividade caia no início. As mudanças não devem ser feitas todas de uma só vez pois isso levaria inevitavelmente ao fracasso do processo. As mudanças têm que ser absorvidas aos poucos pelas organizações.

- **Crie um processo simples**: um novo processo tem que ter sempre como objetivo melhorar a produção de software de forma a melhor atender os clientes da organização. O processo é um meio e não um fim. Não adianta criar algo muito complicado que as pessoas não entendam ou não consigam seguir. O processo precisa resolver os problemas da forma mais simples possível e assim ajudar a melhorar a produção.

- **Lembre-se do objetivo final**: como dito anteriormente, o processo é um meio e não um fim. Lembre-se que o objetivo final é produzir software com mais qualidade e o

menor custo possível.

- **Trate a melhoria de processos como um projeto**: coloque alguém experiente em desenvolvimento de software e em processos como responsável pela definição do processo. Defina os requisitos do processo, modele, implemente e teste em um projeto piloto para depois passar a utilizá-lo em projetos reais.

- **Faça a melhoria de seu processo em ordem de dificuldade**: as práticas apresentadas nos cinco níveis do CMM foram distribuídas de forma a facilitar sua introdução numa organização. Baseie-se nessa ordem para inserir as mudanças em seu processo.

- **Lembre-se que nem democracias completas e nem ditaduras irão funcionar**: é impossível chegar-se a um consenso com relação a todos os aspectos envolvidos em um processo. Por outro lado, ditar regras sem buscar conhecer o trabalho das pessoas e ouvir suas idéias, também não irá ajudar na aceitação do novo processo. É importante buscar consenso em algumas partes, mas outras terão que ser impostas, pelo menos de início.

- **Identifique os produtores e consumidores dos artefatos do processo**: é preciso verificar se tudo que está sendo produzido é realmente necessário, ou seja, se está sendo consumido. Não se pode perder tempo produzindo algo que não será realmente utilizado, ou que não agregue valor algum ao projeto como um todo.

- **Lembre-se que definir o processo é a parte mais fácil**: depois de definir o processo vem o mais difícil, que é fazer com que as pessoas o sigam. É preciso fazer um trabalho constante com as equipes, e proporcionar as mudanças necessárias para que o processo possa ser aplicado. Caso contrário, ele será deixado de lado e todos voltarão a trabalhar da forma como estão acostumados.

- **Aproveite idéias bem sucedidas de outros processos**: é importante estudar e conhecer exemplos de processos bem sucedidos e tirar deles o que têm de melhor. Montar o seu processo a partir de práticas já consolidadas é bem menos arriscado.

- **Crie um grupo de Engenharia de Processos**: este grupo seria responsável pela definição, melhoria e avaliação dos processos em sua organização. Poderia, também, dar suporte quanto a dúvidas com relação aos processos. Deve ser um grupo pequeno e com pessoas com dedicação exclusiva a este trabalho.

- **Documente cada fase do processo no nível adequado**: nem todas as fases precisam ser documentadas com o mesmo nível de detalhe. Tudo depende do número de pessoas que irão lidar com os artefatos de uma fase, o nível de conhecimento dessas pessoas e o nível de dificuldade da fase. Quanto maior o número de pessoas necessárias para trabalhar em uma determinada fase, mais detalhada tem que ser sua documentação.

- **Descreva o processo com o nível de detalhe adequado**: o processo precisa ser descrito até um nível que os profissionais que irão utilizá-lo consigam segui-lo passo a passo. Não deve ser detalhado demais a ponto de tolher os desenvolvedores, nem ser genérico demais a ponto de deixá-los perdidos sem saber executar as tarefas.

- **Faça com que todos sejam responsáveis pela melhoria do processo**: a gerência tem que ser responsável por acompanhar a implantação do processo e sua utilização, dar o suporte necessário e resolver os problemas que forem surgindo. Os desenvolvedores têm que ser responsáveis por aprender e utilizar o novo processo, reportar dificuldades e auxiliar os demais.

- **Consulte um especialista**: melhoria de processo é uma tarefa complexa e difícil. A ajuda de um especialista, com experiência em definição de processos específicos para uma organização, pode facilitar e até ajudar a reduzir os custos da implantação de um novo processo (a longo prazo).

- **Faça com que todos entendam o processo e as técnicas envolvidas**: para que o novo processo tenha sucesso, não basta que as pessoas o apliquem burocraticamente a seu trabalho. Os usuários do processo precisam aprender as técnicas envolvidas e saber os "porquês" de cada fase para que o trabalho torne-se produtivo. O processo não pode se transformar simplesmente em adereço, quer dizer, as pessoas continuem trabalhando como tinham costume e depois cumpram com a "burocracia imposta pelo processo".

- **Crie um guia para o processo**: se a documentação do processo está extensa e difícil de ser seguida, crie um guia rápido para que as pessoas possam começar a se acostumar com os novos procedimentos.

- **Tenha paciência**: ao implantar um novo processo, a produtividade primeiro irá cair para depois de um bom tempo começar a crescer novamente. O completo domínio do processo pelos usuários pode levar anos.

- **Adeque o processo a seus objetivos**: mais uma vez, de nada adianta ter um processo que seja somente um entrave burocrático no projeto. Leve em consideração as características de cada projeto e adeque o processo a eles, dando maior ou menor ênfase às atividades de acordo com as prioridades.

- **Defina seu processo o quanto antes**: quanto mais se demorar a definir um processo, mais difícil e maiores serão as mudanças. Mais arraigado cada um vai estar com o seu jeito de trabalhar e mais difícil vai ser adotar as novas práticas.

Além de todos estes aspectos, é importante lembrar, também, que um processo nunca é perfeito e sempre vai haver pontos a serem melhorados. O processo não pode ser dado como fechado e finalizado, independentemente da experiência de quem o tenha definido. É imprescindível realizar reuniões com os usuários do processo no dia a dia para descobrir defeitos

e obter sugestões de melhorias. Podem ser estabelecidas reuniões de avaliação ao final de cada projeto, ou, ainda, com intervalos regulares de tempo, de forma independente do cronograma do projeto.

Capítulo 5

O WebPraxis

5.1 Introdução

5.1.1 Definições Iniciais

Os produtos a serem desenvolvidos utilizando-se o WebPraxis são aplicações Web. O WebPraxis se destina a aplicativos para rodar em ambiente web[1] e não simplesmente sítios estáticos ou dinâmicos. Tais produtos são softwares feitos sob encomenda, não se tratando de "software de caixinha".

Os principais atributos destes produtos são aqueles considerados críticos no desenvolvimento para a Web:

- **Segurança dos dados**: uma grande preocupação das aplicações Web, hoje, é conseguir garantir a integridade dos dados que transitam pela rede. Essa preocupação aumenta na medida em que estas aplicações evoluem e assumem o controle de áreas consideradas críticas para as empresas ou pessoas físicas, como movimentação bancária, transações comerciais, etc.

- **Facilidade de utilização do produto final**: as aplicações Web são acessadas por uma enorme gama de pessoas, com características muito distintas, com pouca ou nenhuma habilidade para utilizar um computador. É necessário empregar mais recursos em estudos de usabilidade para tentar atender este público tão diverso.

- **Design gráfico atrativo, cativante, mas, ao mesmo tempo, funcional**: além da diversidade do público citada acima, um outro fator deve ser considerado para mostrar a importância de uma aplicação atrativa visualmente e fácil de usar: na Web, a concorrência está a um clique de distância. Seja qual for o serviço que sua aplicação

[1] Como exemplo, pode-se citar aplicações de comércio eletrônico e de educação à distância.

oferece, provavelmente existem outros tantos fornecedores desse mesmo serviço espalhados pelo mundo, mas que seu usuário pode passar a conhecer e utilizar com um simples clique de mouse.

- **Tempo de resposta baixo**: ou seja, bom desempenho da aplicação. Deve-se lembrar que grande parte do público de uma aplicação Web ainda utiliza conexões de velocidade baixa, como via rede telefônica. É importante se preocupar com a performance da aplicação e, principalmente, com o tempo de carga das páginas na máquina dos usuários.

- **Portabilidade**: uma aplicação Web, salvo exceções como a intranet de uma empresa, deve poder ser acessada, sem apresentar problemas, através de diferentes navegadores e em diferentes plataformas. Isto se deve, mais uma vez, ao fator da diversidade do público que utiliza a aplicação.

- **Conteúdo atualizado e dinâmico**: para ser sempre atrativa ao público, a aplicação não pode passar a idéia de que está desatualizada e deve sempre procurar apresentar novidades, tanto com relação ao conteúdo apresentado em suas páginas quanto com relação a funcionalidades. A manutenção do conteúdo deve ser considerada como um aspecto importante desde o início do projeto.

- **Correção e robustez da aplicação**: como numa aplicação tradicional, a aplicação deve se preocupar em fazer corretamente o que se espera que ela faça e se comportar de forma não desastrosa em situações não previstas. O agravante para as aplicações Web é, mais uma vez, a proximidade da concorrência. Erros podem custar a perda de usuários que já eram cativos e que foram difíceis de se conquistar.

Além dos atributos dos produtos a serem gerados pelo processo, é preciso levar em consideração os objetivos do novo processo, o que se pretende alcançar com sua implantação. O WebPraxis tem como objetivo melhorar a produção de aplicações Web. Esse objetivo genérico pode ser traduzido em metas mais específicas, tais como:

- **Facilitar a gerência dos diferentes profissionais envolvidos no projeto**: devido ao caráter híbrido das aplicações Web, já mencionado anteriormente, seu desenvolvimento envolve uma maior variedade de profissionais. Um dos objetivos do WebPraxis é auxiliar na gerência e na integração do trabalho desses profissionais. O simples fato de se ter uma abordagem organizada e bem definida para as atividades que devem ser realizadas, definindo por quem e em que ordem, já é um fator de motivação para a equipe. Para que o profissional se sinta motivado, é muito importante que ele tenha consciência de seu papel dentro das atividades do projeto e que saiba em que o seu trabalho está contribuindo.

- **Minimizar o tempo de disponibilização do produto no mercado**: quando se trata de Web, os prazos para a disponibilização do produto são ainda menores e mais críticos. Isso requer um planejamento muito bem feito para que se possa atender o cliente, mas sem prejudicar a qualidade do que está sendo desenvolvido.

- **Produzir um sistema que se aproxime o máximo possível das expectativas do cliente**: como no desenvolvimento de aplicações tradicionais, é de extrema importância que o cliente fique satisfeito com o resultado do projeto. Isto ocorrerá se, ao longo do projeto, a confiança do cliente no fornecedor não for abalada por prazos e custos estourados e se o produto final atender as necessidades do cliente da forma como ele tinha imaginado. Para se alcançar este objetivo, mais uma vez, o planejamento e o acompanhamento do projeto devem ser muito bem feitos, e a fase de levantamento deve ser bem estruturada e dimensionada.

- **Auxiliar no cumprimento dos prazos e custos estipulados**: não existem fórmulas mágicas para resolver este que é um dos maiores problemas da indústria de software. Para se alcançar um nível de planejamento de custos e prazos que sejam compatíveis com a realidade, é preciso conhecer a força de trabalho da empresa, a produtividade das equipes, é preciso ter uma noção clara da dimensão do projeto, dos riscos e problemas que podem surgir. Atividades como levantamento de requisitos, gestão de riscos e coleta de métricas, são muito importantes para se alcançar tais objetivos. Mas não adianta somente saber estipular prazos e custos, é preciso fazer um acompanhamento de cada uma das etapas do projeto para garantir que estes sejam cumpridos.

- **Minimizar o número de manutenções corretivas**: sabe-se que corrigir problemas depois do sistema pronto e em produção é muito mais difícil e caro do que corrigi-los ainda durante o desenvolvimento. O fato de dividir o sistema em liberações que são desenhadas, implementadas, testadas, revisadas e integradas com as anteriores já é um grande passo no sentido de identificar problemas o quanto antes e corrigi-los enquanto o sistema ainda não é tão grande e complexo. Várias outras atividades auxiliam no cumprimento deste objetivo, tais como a definição bem feita dos requisitos, o desenho robusto e revisado da arquitetura do sistema como um todo e de cada uma de suas partes, os procedimentos de revisão e controle feitos entre as atividades.

- **Possibilitar estimativas de custos e prazos mais próximas à realidade**: como dito anteriormente, é preciso conhecer a força de trabalho da empresa, a produtividade das equipes e se ter uma noção clara da dimensão do projeto. Atividades como levantamento de requisitos, gestão de riscos e coleta de métricas, são muito importantes para se alcançar tais objetivos. É importante formar uma base histórica com as experiências de outros projetos da organização que poderá ser utilizada em projetos futuros. Para formar esta base, é preciso coletar dados sobre as atividades desenvolvidas durante o projeto, o tempo e recursos gastos em cada uma delas.

- **Otimizar a utilização dos recursos disponíveis**: quando se tem um processo de desenvolvimento bem definido e também é feito um planejamento do projeto, essa otimização é possível pois pode ser explorado o paralelismo entre certas atividades.

- **Auxiliar no conhecimento da força de trabalho da empresa**: mais uma vez, a coleta de métricas mostra-se importante. Este objetivo tem que ser atingido para que se possa conseguir também o de estipular custos e prazos mais reais.

- **Aumentar a qualidade do software produzido**: este objetivo é tão genérico que praticamente todas as atividades do processo contribuem para seu cumprimento. Para produzir um software com mais qualidade é preciso definir seus requisitos, produzir um desenho robusto, testar, revisar, corrigir os problemas encontrados. O fato de se ter um processo que comporte estas atividades e que tenha o controle da qualidade como preocupação é imprescindível.

- **Favorecer o reuso**: já foi dito que os prazos para a entrega de produtos para a Web são ainda menores do que para sistemas tradicionais. Sendo assim, nada melhor do que reutilizar componentes já testados e de qualidade. O desenho das várias partes de uma aplicação deve levar sempre em consideração a possibilidade de se reaproveitar o que está sendo feito em versões futuras ou até mesmo em outras aplicações que a organização venha a desenvolver.

- **Gerar produtos de fácil manutenção e expansão**: também como já foi dito, um erro numa aplicação Web, pode custar a perda de usuários difíceis de serem conquistados novamente. Assim, os produtos devem ser feitos de forma a facilitar futuras manutenções, para que elas sejam feitas de forma rápida e com o menor custo possível. Como as aplicações Web devem sempre procurar se manter atualizadas e, na medida do possível, oferecer novas funcionalidades para seus usuários, os produtos devem poder ser facilmente expandidos. O desenho do produto deve ser feito de forma a facilitar estes dois fatores.

Para se classificar as atividades que precisariam ser incluídas no PRAXIS para adequá-lo ao desenvolvimento para Web, foi utilizada a técnica descrita nas seções 4.2.1 e 4.2.2. Os atributos do produto e os objetivos do novo processo foram organizados em ordem de prioridade e relacionados às atividades do processo. Ao final desse procedimento, a cada atividade correspondia um valor numérico indicando seu grau de importância, indicando o quanto aquela atividade contribuía para se suprir os atributos do produto e para se chegar aos objetivos do processo. Neste trabalho, considerou-se que os atributos do produto e os objetivos eram igualmente importantes e por isso, nos cálculos, utilizou-se para ambos o mesmo peso. As tabelas com os resultados dessa classificação, geradas para o WebPraxis, podem ser encontradas na íntegra no Apêndice A, deste documento.

A partir do resultado obtido nesta classificação, foram definidas as atividades necessárias e

partiu-se para distribuir tais atividades entre as fases do WebPraxis.

5.1.2 Visão Geral do Processo

O WebPraxis é um processo de desenvolvimento de software adequado para projetos voltados para a Web, com duração de seis meses a um ano.

Assim como o PRAXIS, no WebPraxis o desenvolvimento é dividido em fases, que são as divisões de caráter gerencial. Cada fase possui uma ou mais iterações. Uma iteração representa um ciclo completo de desenvolvimento, passando por todos os fluxos (divisões de caráter técnico).

O WebPraxis é uma especialização do PRAXIS e, como tal, mantém as fases e os fluxos presentes neste processo, direcionando suas atividades para aspectos específicos do desenvolvimento para a web. Em cada uma das atividades já existentes nos fluxos do PRAXIS, procurou-se dar uma abordagem web, enfatizando aqueles aspectos que não podem deixar de ser avaliados e como isso pode ser feito. Dessa forma, estão presentes no WebPraxis os fluxos do PRAXIS com suas atividades adaptadas, descrevendo o que deve ser feito em cada uma tendo em vista um projeto para a Web. Existem, ainda, aspectos que não estão cobertos pelos fluxos do PRAXIS, mas que precisam estar presentes num processo voltado para a Web. Para cobrir tais aspectos, foram inseridos dois novos fluxos: o fluxo de Criação do Conteúdo e o fluxo de Usabilidade. A criação de fluxos separados para as atividades relacionadas com o conteúdo e os estudos de usabilidade pode ser justificada pela diversidade de profissionais que atuam em cada um deles. Essa divisão de tarefas possibilita a execução das atividades em paralelo, contribuindo para a redução do tempo de desenvolvimento.

Assim, o WebPraxis é constituído pelas fases de Concepção, Elaboração, Construção e Transição. Em cada fase, são desenvolvidas atividades dos fluxos técnicos de Requisitos, Análise, Desenho, Implementação, Testes, Criação de Conteúdo, Usabilidade e dos fluxos gerenciais de Gestão de Projetos e Gestão da Qualidade.

É interessante ressaltar que o WebPraxis procura cobrir todo o ciclo de desenvolvimento de uma aplicação Web. Caso a organização produtora de software seja, também, contratada para hospedar e manter o software em produção depois de pronto, é recomendável incluir uma outra fase após a de Transição. Como sugerido por [Ambler00] para o RUP, esta seria a fase de Operação e trataria do período no qual a aplicação está em produção, sendo utilizado por seus usuários, até que seja criada uma nova versão, ou ela seja descontinuada. As atividades desenvolvidas nesta fase, teriam por objetivo principal manter a aplicação em funcionamento 24 horas por dia, 7 dias por semana, corrigir problemas o mais rápido possível e dar o suporte necessário aos usuários. Antes da aplicação entrar realmente em produção, outras atividades de planejamento e de definição da infra-estrutura necessária deveriam ser realizadas, tais como: definição do hardware necessário para a aplicação, configuração adequada destes servidores, planejamento do suporte ao usuário, alocação e distribuição dos recursos humanos nas atividades de atendimento e monitoração, planejamento dos plantões de monitoramento fora do horário normal de trabalho, etc.

A descrição dos fluxos e fases do WebPraxis é feita nas seções que se seguem.

5.1.3 Formação da equipe de projeto

Uma equipe típica para projetos para a Web envolve profissionais das mais diferentes áreas e perfis, os quais o WebPraxis precisa gerenciar. Pode-se identificar diversas subdivisões na equipe, formando equipes menores com papéis bem definidos. Um mesmo profissional pode assumir mais de um papel em uma ou mais equipe, dependendo de seu perfil e qualificação. As várias equipes que podem ser identificadas correspondem, de certa forma, aos fluxos do processo. Assim, as atividades do fluxo de Requisitos seriam desenvolvidas pela equipe de requisitos. Ao se iniciar o desenho alguns dos profissionais da equipe de Requisitos seriam realocados para a equipe de Desenho, para que o conhecimento sobre o escopo do produto seja mais facilmente disseminado e as dúvidas com relação aos requisitos sejam resolvidas de forma mais rápida. Essa realocação parcial poderia ser feita também com relação às equipes de Implementação, Testes, Criação de Conteúdo e Usabilidade, sempre procurando somar a experiência adquirida a respeito da aplicação com a experiência específica dos especialistas em determinadas atividades.

Os novos profissionais que o WebPraxis precisa passar a gerenciar são, principalmente, pessoas da área de Marketing, que auxiliam no processo de levantamento inicial, e os artistas envolvidos com a produção do conteúdo da aplicação. Segundo [Paula00] e [Burdman99], numa equipe de desenvolvimento para a Web, pode-se destacar os seguintes papéis:

- gerente de projeto
- gerente da qualidade
- designer de multimídia
- designer de interfaces
- redator
- animador
- especialista em vídeo
- especialista em áudio
- engenheiro de software
- programador
- especialista em segurança de dados
- especialista em redes
- especialista em testes

Dependendo do nicho da aplicação e da tecnologia envolvida, podem ainda ser necessários outros profissionais especialistas em, por exemplo, planejamento estratégico, especialista no foco do conteúdo, numa determinada linguagem de programação, etc.

5.2 Detalhes dos Fluxos

O WebPraxis possui todos os fluxos do PRAXIS e ainda dois novos fluxos. Algumas atividades, já existentes no PRAXIS, foram detalhadas com o foco voltado para os aspectos importantes para as aplicações Web.

Nas seções seguintes, são descritas em maiores detalhes aquelas atividades dos fluxos do WebPraxis que diferem do PRAXIS, ou que não estão presentes nele. Uma descrição completa dos fluxos do PRAXIS pode ser encontrada em [Paula01].

5.2.1 Fluxos Técnicos

5.2.1.1 Fluxo de Requisitos

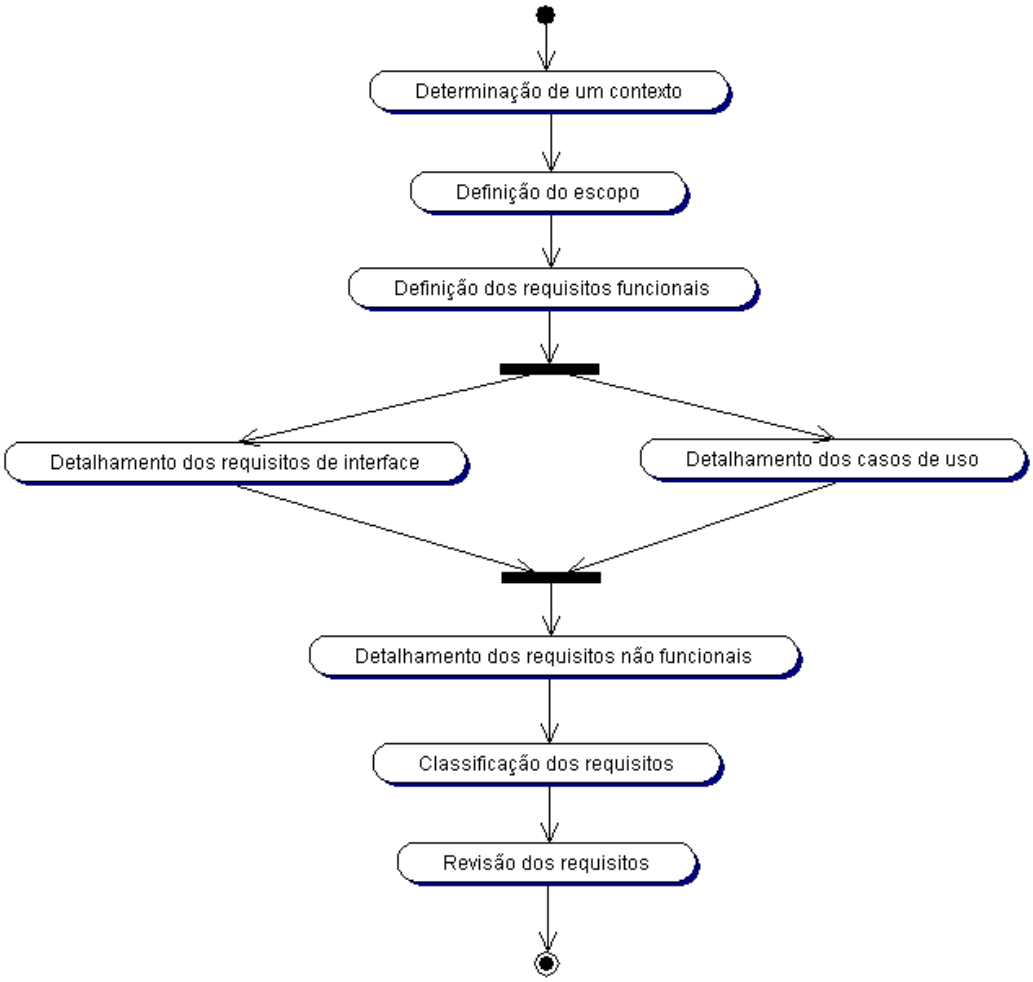

Figura 7 – Atividades do Fluxo de Requisitos

Segundo [Paula01], o fluxo de Requisitos reúne atividades que visam obter o enunciado completo, claro e preciso dos requisitos de um produto de software. No WebPraxis, fazem parte deste fluxo as mesmas atividades do PRAXIS, mas existem alguns aspectos para os quais os analistas precisam estar atentos ao desenvolver cada uma destas atividades e que serão descritos aqui.

Na atividade de Determinação do Contexto da aplicação, deve-se tentar responder as seguintes perguntas: qual o negócio da empresa que está encomendando a aplicação? Onde essa aplicação se encaixa na estratégia da empresa, a que se destina, a quem se destina?

Na segunda atividade, a de Definição do escopo, procura-se delimitar os problemas que o produto se propõe a resolver: que tipo de aplicação web será tratado? Uma intranet, extranet, um sítio institucional dinâmico, uma aplicação de comércio eletrônico, etc. Nessa etapa, também

devem ser definidos quais requisitos serão atendidos numa primeira versão, quantas versões estão previstas e o que vai ser inserido em cada uma delas. Ao encomendar um software, na maioria esmagadora dos casos, o cliente tem muita pressa. No desenvolvimento para a Web, esse fator se agrava. O tempo que o produto leva para ser disponibilizado tem que ser minimizado, caso contrário, ele já é lançado ultrapassado. Uma abordagem que muitos portais e sítios de comércio eletrônico têm adotado é a de colocar no ar uma primeira versão, com as funcionalidades básicas, que pode ser entregue em pouco tempo e ir incrementando o produto já implantado. Essa é uma estratégia que pode ser adotada com sucesso. Pode-se definir com o cliente tudo o que é essencial, e planejar quais outras funcionalidades serão incluídas nas próximas versões. Claro que essa mesma idéia é aplicável no desenvolvimento de qualquer produto, mas no caso da Web, a atualização do produto deve ser uma constante (manter-se atual e atrativo), os prazos de validade de uma versão são menores e o dinamismo, maior.

Na atividade de detalhamento dos requisitos não funcionais, devem ser definidos os requisitos relativos à segurança dos dados, à portabilidade e ao desempenho da aplicação. Um outro aspecto que deve ser explorado nesse ponto é o suporte à personalização planejado para a aplicação.

Por tratar de uma grande variedade de usuários, a personalização é um caminho natural para as aplicações web. Diversas técnicas para a personalização de sítios web têm sido elaboradas, discutidas e testadas [Vieira01]. Tais técnicas são normalmente aplicadas depois que a aplicação já tem um certo tempo de existência no mundo web e já está, de certa forma, com seu funcionamento estabilizado. Alguns autores defendem a idéia de que, antes mesmo da implantação da aplicação, já seja definida uma técnica inicial de personalização a ser utilizada [Fraternali99]. No entanto, essa abordagem pode ser um tanto perigosa, já que inicialmente muito pouco se sabe sobre os usuários que a aplicação pode vir a ter (que podem diferir do público alvo definido inicialmente). Já apresentá-la com uma personalização poderia ser um incômodo para estes usuários e não um diferencial positivo. Recomenda-se, então, definir não a técnica a ser utilizada e sim as informações a respeito das ações do usuário que seriam importantes guardar desde o início, os tão citados *logs* dos usuários. Tais informações poderão ser utilizadas mais tarde para analisar o comportamento dos usuários da aplicação e, então, definir a melhor técnica de personalização a ser empregada.

É necessário se preocupar com a segurança dos dados num projeto para a web em três principais pontos: nas máquinas clientes (as máquinas dos usuários), na máquina servidora, e no tráfego dos dados. No primeiro caso, porque podem ser baixados arquivos com vírus para a máquina do usuário. No segundo, por causa de acessos não permitidos, a dados que não poderiam ser acessados a não ser por determinadas pessoas. No terceiro, porque os dados podem ser alterados no tráfego, fazendo com que as informações sejam perdidas ou corrompidas e também porque pessoas agindo de má fé podem ter acesso a dados que não lhes pertencem, como um número de cartão de crédito, por exemplo. Existem técnicas que podem ser empregadas para resolver o problema de segurança nesses três pontos. A definição de qual ou quais tecnologias serão utilizadas depende do nível de segurança requerido pela aplicação a ser desenvolvida. De acordo com o tipo da aplicação web a ser construída (intranet, extranet, sítio), os requisitos de segurança podem ser mais ou menos rigorosos. É preciso fazer um levantamento preciso, para não deixar o que deve ser protegido desprotegido mas também para não perder tempo “matando

uma formiga com um canhão".

Os aspectos envolvendo a portabilidade da aplicação esbarram no suporte dado pelos diferentes navegadores às tecnologias empregadas na construção das páginas finais da aplicação. Os requisitos de portabilidade também variam. Para uma aplicação disponibilizada para o público em geral, provavelmente definir-se-ia um requisito de que ela deve funcionar perfeitamente em dois ou três tipos de navegador mais utilizados. Para uma intranet, por exemplo, num ambiente conhecido pela empresa, controlado por ela[1], pode ser imposto, então, que todos utilizem um determinado navegador para abrir suas páginas e, então, o requisito de portabilidade não seria tão crucial.

Nos requisitos de desempenho, deve-se determinar uma margem de tolerância para o tempo de carga das páginas. O processamento da maioria das aplicações para a Web é relativamente simples e um dos aspectos que mais influenciam no tempo de carga é o peso das imagens utilizadas. Dessa forma, o requisito de desempenho está intimamente ligado ao trabalho feito no fluxo do Criação do Conteúdo. O equilíbrio entre beleza e desempenho deve estar sempre na mente dos desenvolvedores.

As demais atividades são igualmente importantes e são realizadas de forma semelhante ao PRAXIS.

5.2.1.2 Fluxo de Análise

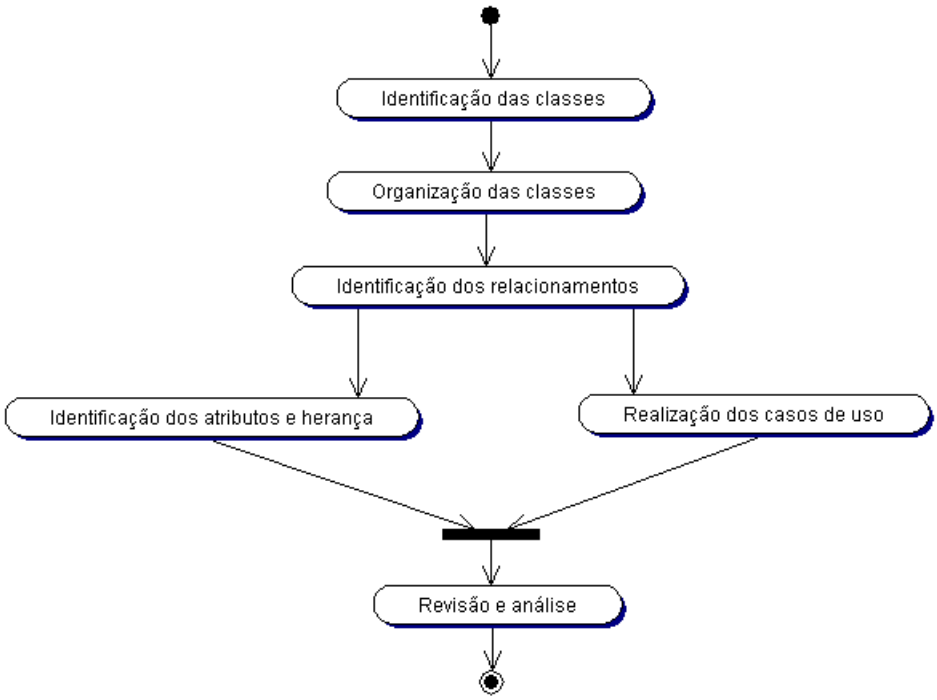

Figura 8 – Atividades do Fluxo de Análise

O fluxo de análise permanece praticamente inalterado com relação ao PRAXIS. A modelagem gerada pela análise deve ser independente de arquitetura ou da tecnologia utilizada. O ideal é que o resultado da análise possa ser utilizado para a implementação do sistema em

[1] É comum a empresa padronizar a utilização de seus softwares, e provavelmente a do navegador também.

qualquer arquitetura que se deseje. Por isso, a análise de um sistema tradicional e de um sistema para a Web pode, inclusive, ser a mesma. As mudanças no nível de modelagem serão sentidas somente no desenho. As classes de fronteira identificadas na análise evoluirão de forma diferente no desenho para uma aplicação tradicional e para uma aplicação Web. No caso da Web, elas serão mapeadas para páginas Web, que são as responsáveis pela interface com os usuários. No caso de uma aplicação tradicional, esse mapeamento dependerá da linguagem de implementação escolhida.

Sendo assim, as atividades para esse fluxo devem ser realizadas da mesma forma como descritas no PRAXIS.

5.2.1.3 Fluxo de Desenho

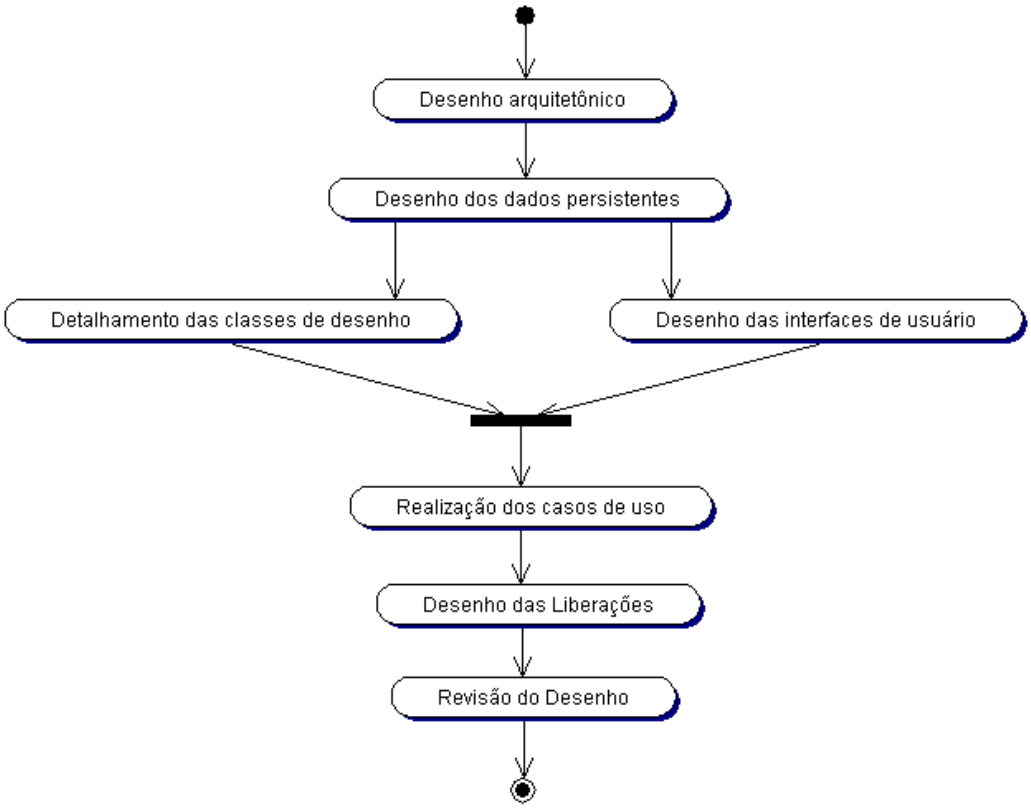

Figura 9 – Atividades do Fluxo de Desenho

Segundo [Paula01], o fluxo de desenho tem por objetivo definir uma estrutura implementável para um produto de software, que atenda aos requisitos especificados para ele. Para se atingir este objetivo, devem ser realizadas as atividades mostradas na Figura 9. As atividades para o fluxo de Desenho no WebPraxis são as mesmas daquelas definidas para o PRAXIS, mas devem ser especializadas para o ambiente Web.

Na atividade de Desenho Arquitetônico é preciso definir qual será a arquitetura adotada para a aplicação a ser construída, assumindo que a arquitetura Web já tenha sido escolhida. Existem diversos padrões (*patterns*) de arquiteturas para aplicações Web que podem ser adotados. Em [Conallen99b], são apresentados três destes padrões, considerados os mais utilizados hoje[1]:

[1] Traduções e definições propostas por Wilson de Pádua, em comunicação particular, para *Thin Web Client*, *Thick Web Client* e *Web Delivery*, respectivamente.

- **Cliente delgado**: os elementos de software estão presentes somente do lado do servidor, sendo as páginas cliente, normalmente, de HTML pura. Ex.: tecnologia JSP, que utiliza *scripts* em *Java* (não *JavaScript*).

- **Cliente espesso**: os elementos de software estão presentes do lado do cliente, podendo estar presentes ou não do lado do servidor. Ex.: *scripts* interpretados pelos navegadores, *applets*.

- **Objetos distribuídos**: neste caso, a tecnologia da Web é utilizada apenas para a entrega de componentes ou aplicativos independentes que, uma vez instalados no cliente, passam a comunicar-se diretamente com o servidor, através de uma tecnologia de objetos distribuídos (por exemplo, RMI ou CORBA.)

Cada um destes padrões possui vantagens e desvantagens. Com o cliente espesso, o processamento fica a cargo das máquinas clientes, o que pode ser um problema já que, na maior parte dos casos, não se sabe qual a configuração das máquinas dos usuários que estarão utilizando uma aplicação Web. Por outro lado, um cliente delgado pode significar um tráfego intenso entre cliente e servidor, o que pode ser problema se as linhas de comunicação forem de baixa velocidade. Deve-se, então, definir qual padrão é o mais apropriado para a aplicação em questão, de acordo com os requisitos levantados e a experiência da organização. Ao final do Desenho Arquitetônico, o produto estará dividido em subsistemas e as tecnologias a serem utilizadas na implementação já deverão estar definidas.

Na atividade de Desenho dos dados persistentes, serão tratadas as estruturas externas de armazenamento de dados, como arquivos e bancos de dados. O principal problema a ser resolvido é a tradução do modelo de desenho orientado a objetos para os paradigmas das estruturas de armazenamento. As estruturas de armazenamento mais utilizadas são os bancos de dados e o paradigma ainda predominante para aplicações comerciais é o relacional. Existem regras para se mapear as classes persistentes do seu modelo de classes para tabelas de um banco relacional. Tais regras já foram alvo de diversos trabalhos, podendo ser extraídas de [Blaha+98a], [Blaha+99], [Brown+97], [Deboni97] e [Rational99]. Além da tradução das classes persistentes para a estrutura de armazenamento, é necessário definir o mecanismo que irá fazer o acesso aos dados armazenados, ou seja, a camada de persistência da aplicação. Segundo [Ambler99c], podem ser distinguidos três tipos de mecanismos de acesso:

- Sem uma camada de persistência especializada, onde os comandos SQL estão inseridos no código fonte das classes de domínio.
- Com a camada de persistência formada por um conjunto de classes que encapsulam o acesso ao banco
- Com a camada de persistência formada por uma estrutura mais elaborada, como se fosse um outro sistema. Esse sistema recebe os objetos e os mapeia para o mecanismo de persistência utilizado.

A construção de uma camada de persistência robusta[1] pode ser trabalhosa e onerar o projeto num primeiro momento, mas seu custo deve, na verdade, ser diluído para todos os projetos a serem desenvolvidos pela empresa posteriormente. Construída da forma sugerida em [Ambler99c], a camada de persistência independe do mecanismo de armazenagem utilizado no projeto e é totalmente separada do código do projeto. Assim, ela pode ser utilizada posteriormente em todos os projetos da organização. Uma outra vantagem é que o acesso aos dados fica transparente aos desenvolvedores, que não precisam se preocupar em dominar ou codificar o acesso ao mecanismo de armazenamento.

Nas atividades de Detalhamento das classes de desenho e Realização dos casos de uso serão explicitadas as diferenças entre a modelagem para uma aplicação tradicional e uma aplicação Web. Nestas atividades, será utilizada a extensão para a UML descrita no Capítulo 3. As classes do modelo de análise serão detalhadas e modeladas utilizando-se os novos estereótipos introduzidos pela WAE para aspectos específicos de páginas web. Por exemplo, as classes de fronteira da análise são mapeadas para páginas cliente, as classes de controle para páginas servidoras.

O Desenho das interfaces de usuário deve ser feito em um nível de detalhe maior do que no fluxo de Requisitos, já se levando em consideração a tecnologia a ser utilizada na implementação das mesmas. Em uma aplicação Web, as páginas são, em última análise, arquivos HTML. Deve-se procurar utilizar os recursos padronizados, que são suportados pelos diferentes navegadores. Deve-se evitar extensões HTML que só poderão ser interpretadas se o usuário utiliza um tipo específico de navegador.

O Desenho das Liberações define como a construção do produto será dividida em liberações. A Revisão do Desenho valida o esforço realizado nas atividades deste fluxo, verificando se estão compatíveis com os resultados dos Requisitos e Análise.

[1] Termo utilizado por Ambler para designar o terceiro tipo de mecanismo de acesso.

5.2.1.4 Fluxo de Implementação

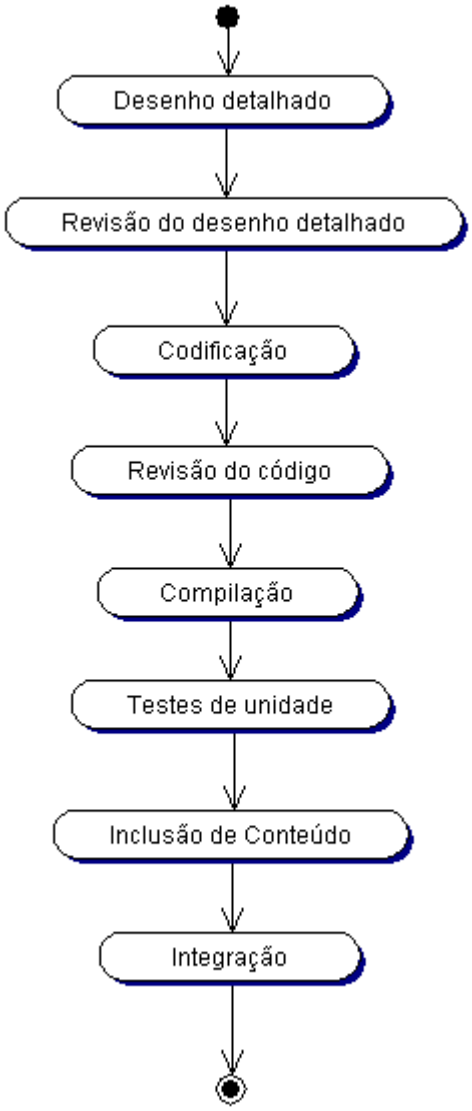

Figura 10 – Atividades do Fluxo de Implementação

No fluxo de Implementação, além das atividades já presentes no PRAXIS, deve ser realizada uma nova atividade: a de Inclusão do Conteúdo. Essa atividade é o principal elo de integração entre o fluxo de Criação de Conteúdo e o fluxo de Implementação. A equipe responsável pela criação do conteúdo para a aplicação irá disponibilizar os arquivos com os itens de conteúdo (textos, vídeos, animações, imagens já no formato e tamanho corretos, arquivos de som) que devem ser incorporados ao código. Depois de inserido o conteúdo, devem ser refeitos os testes de unidade para ver se não foram introduzidos defeitos e se os aspectos de visualização do material estão corretos.

A atividade de Codificação também deve sofrer uma pequena alteração, pois é preciso traduzir o desenho gráfico criado para as páginas. O resultado dessa tradução é um esqueleto das páginas, armazenados em arquivos HTML. Esse esqueleto será utilizado pelos desenvolvedores como molde. Se uma determinada página é gerada dinamicamente, o desenvolvedor sabe que aquele é o aspecto final que ela deve ter e deverá fazer com que ela seja gerada exatamente

assim. Os testes de unidade devem incluir a comparação entre o aspecto das páginas no desenho gráfico e o aspecto das páginas geradas pela aplicação.

5.2.1.5 Fluxo de Testes

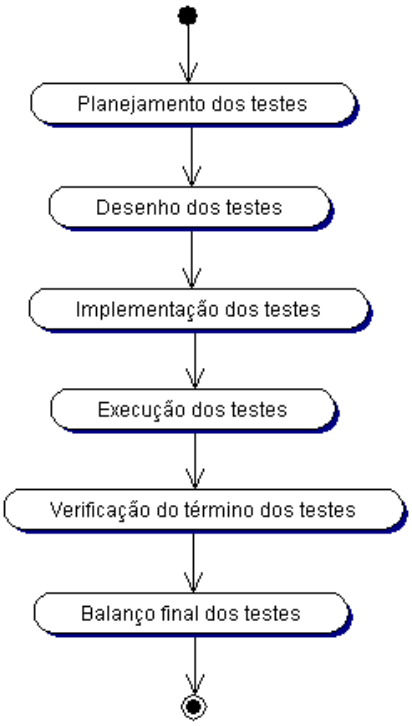

Figura 11 – Atividades do Fluxo de Testes

As atividades do fluxo de teste são as mesmas descritas no PRAXIS, mas alguns aspectos importantes devem ser, aqui, ressaltados [Burdman99]:

- É preciso fazer o planejamento dos testes já no início do projeto. Neste planejamento, devem ser previstos testes para componentes prontos a serem adquiridos e utilizados e testes com novas tecnologias a serem utilizadas pela primeira vez, além dos testes do que será produzido pela equipe.

- Para que o resultado dos testes seja facilmente mensurado pela equipe de testes, os requisitos definidos para o sistema devem estabelecer métricas precisas, tanto quanto possível. Por exemplo, para o requisito de desempenho, pode-se estabelecer que todas as imagens devem ser carregadas em até 5 segundos. Essas métricas precisam ser estabelecidas para todos os requisitos.

- Um outro aspecto imprescindível para que os resultados dos testes sejam efetivos é que o ambiente de testes espelhe o ambiente de produção. Isso parece óbvio, mas muitas vezes as organizações não tomam o devido cuidado, deixando, por exemplo, servidores de produção e desenvolvimento com diferentes configurações. Esta diferença simplesmente invalida os resultados dos testes, pois o que funciona em um

ambiente não necessariamente funciona no outro.

- Ao se passar de aplicações *desktop* para aplicações cliente/servidor e, posteriormente, para aplicações Web, gradualmente se perde o controle do ambiente onde o software será utilizado. Este fato afeta os procedimentos de teste, que devem tentar abranger os diferentes tipos de ambientes.

- A execução de testes de carga é muito importante para aplicações para a Web. Este tipo de teste poderá identificar se o dimensionamento feito para o servidor está correto, se as estruturas e a arquitetura definida para a aplicação comportam uma carga mais pesada sem comprometer o desempenho.

5.2.1.6 Fluxo de Criação de Conteúdo

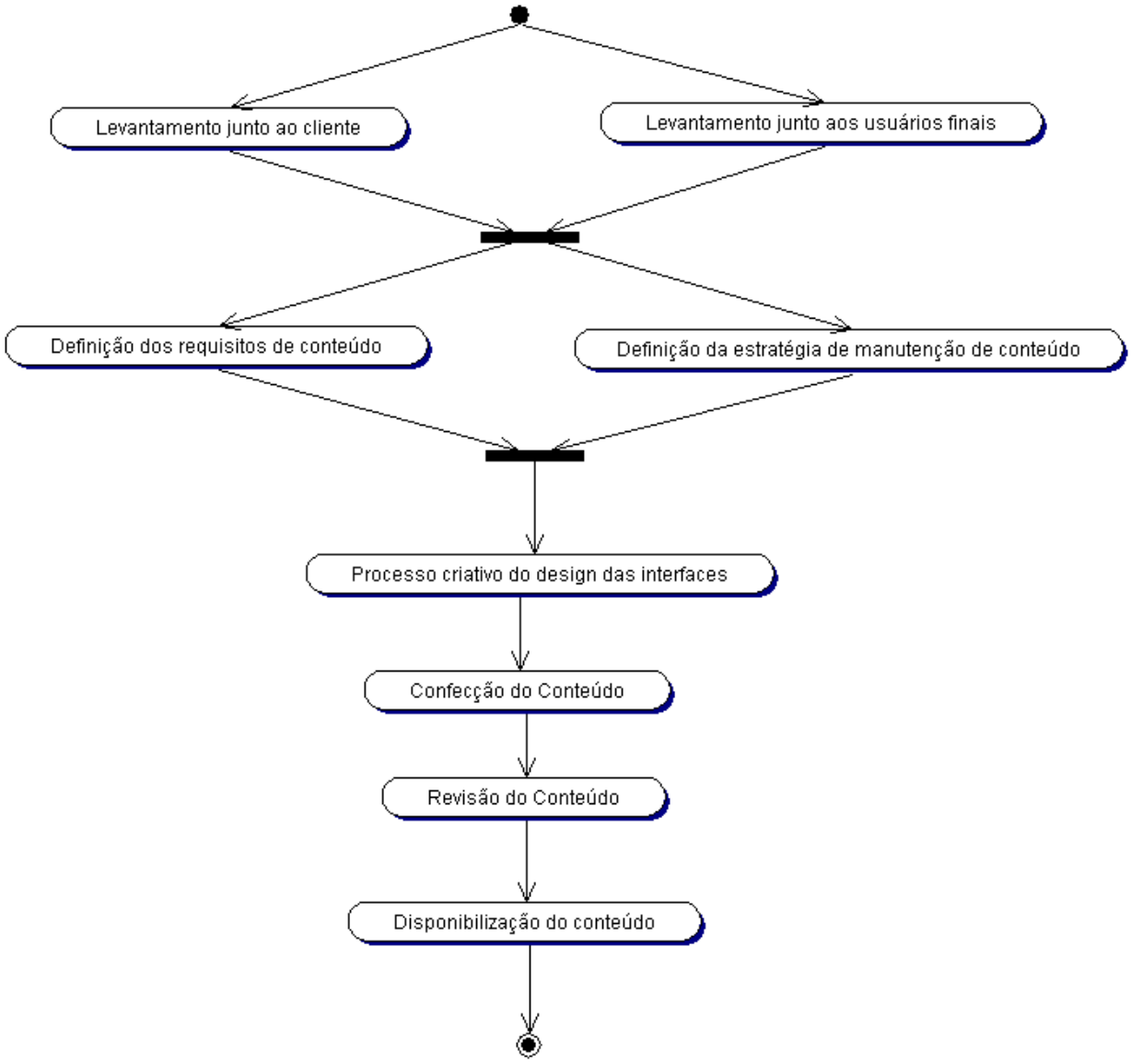

Figura 12 – Atividades do Fluxo de Criação de Conteúdo

O fluxo de Criação de Conteúdo foi inserido no processo para tratar de um aspecto de extrema importância para aplicações web: o conteúdo das páginas. O conteúdo refere-se às imagens, animações, vídeos, textos, hipertexto e sons presentes nas páginas web. Todo esse material deve ser produzido e, posteriormente, quando o sistema entrar em produção, gerenciado. A fase de criação deve preocupar-se em gerar um conteúdo atrativo, agradável e funcional para o público alvo. O conteúdo não deve comprometer o desempenho das páginas. A fase de gerência do conteúdo deve preocupar-se em mantê-lo sempre atualizado e com as características citadas para a criação.

A primeira etapa a ser cumprida neste fluxo é a de levantamento, para que se possa definir, posteriormente, os requisitos de conteúdo. Este levantamento inicial possibilitará a formação do "conceito"[1] do sítio. Um dos objetivos dessa etapa é conhecer o usuário final do produto, para tentar definir o que é interessante e atrativo para ele. Um outro objetivo é conhecer o ponto de vista e objetivos do cliente, aquele que encomendou o produto. Para atingir tais objetivos, são realizadas duas atividades:

- **Levantamento junto ao cliente**: o conteúdo da aplicação a ser desenvolvida deve estar de acordo com as estratégias de marketing da empresa. Na maioria dos casos, a disponibilização de uma aplicação na Web envolve profissionais da área de marketing devido a grande exposição que a Internet proporciona para a empresa. Mesmo que o cliente não possua uma equipe de marketing ou estratégias pré-estabelecidas, é importante levantar junto dele o que ele espera da aplicação, quais são seus objetivos, que tipo de público se pretende atingir, etc. É preciso levantar, também, se já existe alguma pré-disposição com relação a cores a serem utilizadas, tipos de objetos gráficos, elementos multimídia (vídeos, sons, animações) e o enfoque a ser dado nos textos. A equipe que irá trabalhar o conteúdo da aplicação precisa definir junto ao cliente em quais pontos ela deverá criar partindo do zero e em quais pontos ele já possui uma idéia do que quer que seja feito.

- **Levantamento junto aos usuários finais**: para este levantamento, podem ser utilizadas pesquisas de marketing encomendadas para alguma agência especializada ou podem ser conduzidas reuniões com grupos de usuários[2]. Nessas reuniões, representantes dos diferentes grupos de usuários e a equipe de conteúdo interagem para determinar as necessidades e características dos usuários. Podem ser feitas enquetes com perguntas como sítios que acham interessantes, cores, o que consideram importante em um sítio[3], etc. Dependendo do porte do cliente, muitas vezes esse levantamento já terá sido feito pela equipe de marketing dele, ou por alguma empresa que ele tenha contratado anteriormente. Nesse caso, os resultados do levantamento devem ser repassados para a equipe de conteúdo para que ela possa passar para a etapa seguinte.

[1] Termo usado em Marketing para designar a linha que se deve seguir, uma idéia geral com relação ao produto.
[2] *Focus groups.*
[3] A referência sempre a sítios se justifica pois do ponto de vista dos usuários uma aplicação web não passa de um sítio. O processamento feito por trás é transparente para ele.

A partir dos resultados das duas primeiras atividades, devem ser definidos os requisitos de conteúdo para a aplicação. Com relação à parte gráfica, devem ser definidas as cores predominantes, formatos a serem utilizados (arredondados, ovais, etc.), o estilo a ser seguido, por exemplo, se as páginas serão mais textuais ou icônicas. Com relação aos textos, devem ser definidos fontes, cores e leiautes. É preciso definir, ainda, que tipos de recursos multimídia serão utilizados e quem ficará responsável por eles. Muitas vezes, o cliente já possui algum vídeo, ou imagem que quer utilizar; outras, esse material precisa ser também produzido ou adquirido. No caso de se optar por adquirir parte desse material, por exemplo, imagens de um banco de imagens, o trabalho de busca e seleção deve ser contabilizado.

Em paralelo à atividade de definição dos requisitos do conteúdo, deve ser realizada a atividade de definição da estratégia de manutenção do conteúdo. É preciso definir quem ficará encarregado desta manutenção depois que o software entrar em produção. Tanto no caso do cliente ficar responsável pela manutenção quanto no caso dele contratar os seus serviços de manutenção, é preciso definir como ela será realizada. Fatores como tamanho da aplicação, quantidade de elementos a serem mantidos (quantos vídeos, animações, imagens, textos), complexidade do conteúdo desses elementos, devem ser analisados. Com esta análise, poderá ser definida qual a melhor opção para a manutenção: ser feita através de uma ferramenta de manutenção de conteúdo; ser feita através de um módulo gerencial do sítio, que precisará também ser construído, ser feita diretamente nos arquivos onde está armazenado o conteúdo.

A próxima atividade a ser realizada é o processo criativo do desenho das interfaces. O profissional de projeto gráfico elabora um esboço para as diferentes páginas da aplicação, de acordo com os requisitos levantados. Como costuma ser feito numa campanha publicitária, podem ser elaborados leiautes diferentes, para que o cliente opte por um deles. Feito o esboço, deve-se definir todo o conteúdo adicional a ser incluído em cada página, de forma genérica. Por exemplo, na página inicial, serão incluídos uma animação de abertura, um texto explicativo, um vídeo, e hiperligações para as outras páginas no formato de ícones. Após esse processo criativo, é necessário obter a aprovação do cliente para se passar para a etapa seguinte de confecção de todo o material a ser utilizado.

A atividade de confecção do conteúdo pode ser dividida em atividades menores, descritas a seguir [Paula00]:

- **produção dos textos**: redação, revisão e formatação (conforme o especificado anteriormente)
- **produção do material**: captura do material (gravação de áudio e vídeo, tomada de fotos, busca e seleção de imagens a serem compradas, digitalização do material), edição do material e pós-processamento (retoques, efeitos especiais, combinação de materiais)

Após a confecção, todo o conteúdo deve ser revisto para verificar se está de acordo com as especificações iniciais e se o nível de qualidade atingido é satisfatório. É importante que a revisão seja feita por especialistas e também pelo cliente. Os especialistas devem se preocupar com, por exemplo, correção dos textos quanto à linguagem, tamanho e definição adequados dos arquivos de imagens, som e vídeo, etc. A revisão do cliente tem por objetivo apontar aspectos que não estão de acordo com o que foi pedido, talvez por imprecisão dos requisitos. Após corrigir os

problemas encontrados, é necessária a aprovação por parte do cliente para se passar para a atividade seguinte.

Depois de criar o conteúdo e revisá-lo, o próximo passo é disponibilizá-lo para a equipe de desenvolvimento para que possa ser integrado ao código das páginas. [Burdman99] sugere um método simples para gerenciar o que já está disponível para a equipe de desenvolvimento: criar duas estruturas de diretório semelhantes, uma onde está armazenado todo o conteúdo produzido e com acesso restrito para a equipe de conteúdo e outra onde vão sendo copiados os arquivos já prontos e revisados e que a equipe de desenvolvimento tenha acesso. Com relação ao desenho gráfico das páginas, alguns passos diferentes devem ser cumpridos. É preciso definir quem ficará responsável pela tradução do conteúdo de um arquivo gráfico, por exemplo criado no *Photoshop*, para uma página real, codificada em HTML. Para realizar esta tradução, é necessário fazer o recorte das imagens e, depois, estruturá-las no código HTML. Essa atividade de recorte das imagens e a de montagem de um esqueleto para as páginas deve ser feita em conjunto pelo profissional do desenho gráfico e pelos desenvolvedores, pois irá depender da estrutura definida para a construção das páginas, como a utilização de *frames* ou não, a definição de que estrutura estará por baixo para receber as imagens. Esse esqueleto das páginas será utilizado pelos desenvolvedores como molde. Se uma determinada página é gerada dinamicamente, o desenvolvedor sabe que aquele é o aspecto final que ela deve ter e deverá fazer com que ela seja gerada exatamente assim.

5.2.1.7 Fluxo de Usabilidade

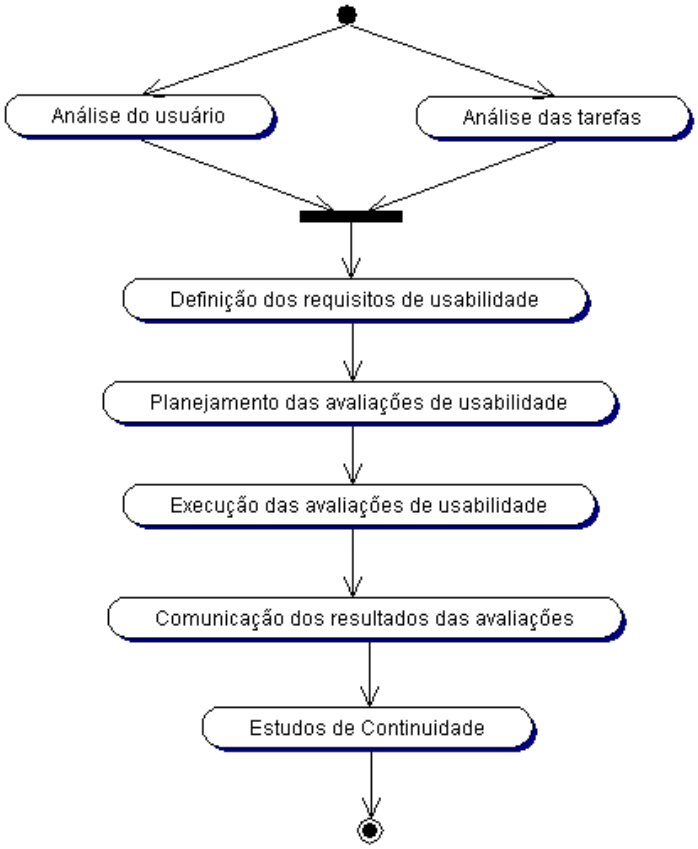

Figura 13 – Atividades do Fluxo de Usabilidade

As atividades do fluxo de Usabilidade foram propostas por [Ribeiro99] como uma extensão para o PROSE, processo que antecede o PRAXIS. Tais atividades podem ser inseridas no WebPraxis, podendo ser desenvolvidas em paralelo com outras atividades dos outros fluxos.

Segundo [Nielsen00], existem diversos aspectos que devem ser considerados com relação à usabilidade de páginas Web. Dentre eles, destacam-se:

- a grande maioria dos usuários não utiliza as barras de rolagem, projete suas páginas utilizando o espaço da tela, colocando o que for mais importante nessa área.
- evite utilizar muitos *frames* e novas instâncias do navegador em suas páginas, esses recursos muitas vezes só confundem os usuários.
- ao escrever textos para a Web, seja sucinto. Os textos devem ser escritos especialmente para a Web, aproveitando-se o recurso de hiperligações para melhor estruturá-lo. Não se deve simplesmente copiar um texto impresso para uma página Web. As páginas são um novo paradigma de comunicação e o conteúdo a ser disponibilizado deve ser preparado especialmente para elas.
- minimize o tempo de carga das páginas, esse é um dos maiores problemas da usabilidade na Web.
- procure utilizar os recursos visuais de forma consistente com os grandes sítios, ou com o que é comumente encontrado na Web. Por exemplo, a indicação de que uma palavra é uma hiperligação por estar sublinhada, o ícone de uma seta para a esquerda para voltar à página anterior, etc.
- mantenha o desenho gráfico simples. A parafernália visual de uma aplicação Web não deve mascarar a verdadeira missão do sítio: disponibilizar um serviço para o usuário. Certamente, terá mais sucesso um sítio com desenho gráfico simples, mas onde os usuários conseguem encontrar o que desejam e realizar as tarefas que precisam de forma fácil, rápida e correta.
- é imprescindível realizar avaliações de usabilidade com usuários para descobrir falhas, para verificar se existe alguma dificuldade grave na utilização do sítio.

Levando-se tais aspectos em consideração, devem ser realizadas as atividades mostradas na Figura 13.

Na atividade de Análise dos Usuários, o objetivo é obter um conjunto de definições de classes de usuários. Para isto, podem ser analisados os resultados de pesquisas de marketing, questionários, estudos de observação no futuro local de implantação do sistema ou entrevistas com usuários especialistas no domínio da aplicação. Podem, ainda, ser feitas reuniões e grupos de discussão com desenvolvedores e representantes dos usuários, para se tentar determinar as características e necessidades dos futuros usuários.

Na atividade de Análise das Tarefas, o objetivo é buscar conhecer e caracterizar a forma como os usuários desempenham atualmente as tarefas que a aplicação se propõe a implementar, as suas necessidades de informação, como procedem em uma situação de exceção ou emergência e quais são seus objetivos gerais ao realizar estas tarefas.

Com base no levantamento feito nas duas atividades anteriores, são definidos os requisitos de usabilidade. Para auxiliar na definição dos requisitos, podem, ainda, ser feitas sessões JAD e

protótipos descartáveis para serem analisados junto aos usuários. Definidos os requisitos, eles devem ser classificados em ordem de prioridade, pois pode ser difícil atender a todos numa primeira versão. Segundo [Ribeiro99], os requisitos de usabilidade devem conter um atributo de usabilidade (característica a ser avaliada), o instrumento de medida (como os valores para a avaliação dessa característica serão obtidos), o valor a ser medido, o nível atual desse valor, o pior nível aceitável, o nível planejado e o melhor nível possível.

Depois de definidos os requisitos, deve ser feito o planejamento das avaliações de usabilidade. Para este planejamento, deve ser feita uma análise de custo/benefício das avaliações, levando-se em consideração fatores como pessoas-hora a serem gastas na preparação e realização de cada avaliação, pessoas-hora gastas nas modificações sugeridas depois das avaliações, pessoas-hora dos usuários de teste, horas de uso do laboratório de usabilidade e a metodologia de avaliação a ser adotada. Com base nessa análise, será estabelecido o número de avaliações a serem feitas e em quais iterações elas serão realizadas. Deve ser estabelecido um cronograma para as avaliações, com a distribuição de tarefas e alocação dos recursos necessários.

A próxima atividade deste fluxo é a execução das avaliações de usabilidade. Uma avaliação de usabilidade pode ser um estudo com usuários reais ou uma avaliação feita por especialistas segundo um conjunto de regras bem definidas. Seja qual for o método utilizado, ao final da avaliação, deve-se fazer uma análise dos dados e depois um levantamento dos problemas de usabilidade encontrados. A lista dos problemas é, então, submetida a uma revisão pela equipe de usabilidade e desenvolvedores para se determinar a seriedade dos problemas, possíveis soluções e uma estimativa de custo para implementar a solução.

A atividade chamada de Estudos de Continuidade é realizada quando o sistema já está pronto e entrando em produção, ou seja, geralmente a partir da Operação Piloto. O objetivo dessa atividade é obter informações de usabilidade para uma próxima versão do produto. As principais técnicas utilizadas são estudos de marketing, estudos de campo, instrumentação de código e análise do histórico de manutenções e das sessões de suporte aos usuários.

5.2.2 Fluxos Gerenciais

Os fluxos gerenciais do PRAXIS foram criados baseando-se nas práticas do CMM nível 2, sendo eles: o fluxo de Gestão de Projetos e o fluxo de Gestão da Qualidade. No fluxo de Gestão de Projetos, são realizados os procedimentos de gestão dos requisitos, de planejamento do projeto, de controle do projeto e de subcontratação. No fluxo de Gestão da Qualidade, são tratados os aspectos referentes à garantia da qualidade (auditorias da qualidade), à gestão de configurações, ao processo de manutenção do software e às atividades de gestão de revisões.

Para o WebPraxis, permanecem estes dois fluxos e devem ser realizadas todas suas atividades, só que, agora, levando-se em consideração os novos artefatos e serem produzidos e suas implicações. Por exemplo, na gestão dos requisitos, devem ser cadastrados, monitorados e rastreados os requisitos de conteúdo e de usabilidade. Os artefatos produzidos pelas atividades destes dois novos fluxos devem ser também revisados e submetidos aos procedimentos de controle já existentes. No planejamento do projeto, devem ser considerados os recursos necessários para se desenvolver as novas atividades.

Uma explicação detalhada a respeito das atividades dos fluxos gerenciais do PRAXIS pode

ser encontrada em [Paula01].

5.3 Detalhes das fases

As fases do WebPraxis correspondem às fases do PRAXIS, sendo mostradas a seguir. As principais modificações ficam a cargo dos dois novos fluxos acrescentados no WebPraxis, que inserem, em cada fase, novas atividades a serem realizadas pela equipe do projeto. Para cada fase, são apresentados *scripts*, baseados nos scripts do PRAXIS [Paula01], que servem de guia para os gerentes de projeto. Tais *scripts* mostram de forma sucinta os pré-requisitos, condições de término, os artefatos consumidos e produzidos, os fluxos e as suas atividades a serem realizadas em uma determinada iteração de uma fase.

Nas seções seguintes, são descritos, em maiores detalhes, aqueles aspectos nas fases do WebPraxis que diferem do PRAXIS. Uma descrição completa das fases do PRAXIS pode ser encontrada em [Paula01].

5.3.1 Concepção

5.3.1.1 Ativação

A fase de Concepção possui uma única iteração: a Ativação. O objetivo desta iteração é verificar se o cliente tem necessidades de negócio suficientes que justifiquem a elaboração de uma especificação de um produto de software, que seria feita na fase de Elaboração. Como, muitas vezes, esta fase não é cobrada do cliente, deve ser o mais breve possível. Deve-se procurar fazer um levantamento grosseiro do escopo e viabilidade de uma idéia de produto, mas que seja suficiente para dimensionar a fase de Elaboração. O *script* para a Ativação é mostrado na Tabela 1.

As atividades mais importantes da Ativação fazem parte do fluxo de Requisitos, do fluxo de Criação de Conteúdo, do fluxo de Usabilidade e do fluxo de Gestão. No fluxo de requisitos, será definido o escopo do produto e será realizado um levantamento preliminar dos requisitos. Estas atividades são importantes para que se possa ter uma noção da dimensão do projeto e, assim, tentar medir os esforços necessários para sua realização. As atividades dos fluxos de Criação de Conteúdo e de Usabilidade irão fornecer outros dados para que essa primeira noção seja mais próxima à realidade. O levantamento preliminar dos requisitos de conteúdo deve definir quem ficará encarregado de confeccionar o material e de sua manutenção. Essa decisão implicará em diferentes atividades e custos. Se o cliente, ou outra organização contratada por ele, for ser responsável pelo conteúdo, o trabalho da organização produtora de software restringe-se a determinar prazos para a entrega do material que não comprometam seu cronograma. Caso o cliente queira contratar também a confecção do material, nesta fase de Ativação, deve-se levantar o escopo do conteúdo, o nível de aprofundamento desejado, as tecnologias envolvidas. Estas informações serão cruciais para determinar se deverão ser contratados especialistas e o tempo necessário para elaborar o conteúdo. Em aplicações Web, muitas vezes, o cliente não é

especialista no assunto que envolve a aplicação e não será capaz de fornecer todas as informações necessárias para confecção do conteúdo, sendo necessária uma pesquisa mais detalhada.

As atividades do fluxo de Usabilidade, nesta fase, têm como objetivo determinar se o público alvo requer estudos mais específicos e aprofundados ou, ainda, qual grau de importância que o próprio cliente dá para a Usabilidade. Se a aplicação se destina ao público infantil, por exemplo, ou a pessoas com pouquíssima proficiência com computadores, o tempo e esforços gastos nas atividades posteriores do fluxo de Usabilidade deverão, certamente, ser maiores. O mesmo acontece se for objetivo do cliente desenvolver uma aplicação que atraia o público de seus concorrentes por ser mais fácil de usar e compreender.

<table>
<tr><td>Descrição</td><td colspan="3">Levantamento e análise das necessidades dos usuários e conceitos da aplicação, em nível de detalhe suficiente para justificar a especificação de um produto de software.</td></tr>
<tr><td>Pré-requisitos</td><td colspan="3">Solicitação de proposta.</td></tr>
<tr><td>Insumos</td><td colspan="3">(somente documentos externos ao projeto).</td></tr>
<tr><td rowspan="9">Atividades</td><td>Fluxo</td><td colspan="2">Tarefas</td></tr>
<tr><td>Requisitos</td><td colspan="2">Definição do escopo do produto.
Definição dos requisitos (preliminar).</td></tr>
<tr><td>Análise</td><td colspan="2">Estudos de viabilidade (opcional).</td></tr>
<tr><td>Desenho</td><td colspan="2">Esboço da arquitetura do produto (opcional).
Estudos de viabilidade (opcional).</td></tr>
<tr><td>Implementação</td><td colspan="2">Prototipagem dos requisitos (opcional).</td></tr>
<tr><td>Testes</td><td colspan="2">Testes dos protótipos dos requisitos (opcional).</td></tr>
<tr><td>Criação de Conteúdo</td><td colspan="2">Levantamento junto ao cliente (preliminar).
Definição dos requisitos de conteúdo (preliminar).
Definição da estratégia de manutenção de conteúdo (preliminar).</td></tr>
<tr><td>Usabilidade</td><td colspan="2">Análise do usuário/tarefas (preliminar).
Definição dos requisitos de usabilidade (preliminar).</td></tr>
<tr><td>Gestão</td><td colspan="2">Levantamento das metas gerenciais.
Estimativas da fase de Elaboração.
Elaboração de proposta de especificação.</td></tr>
<tr><td rowspan="2">Resultados</td><td colspan="2">Artefato</td><td>Sigla</td><td>Partes</td></tr>
<tr><td colspan="2">Proposta de Especificação de Software</td><td>PESw</td><td></td></tr>
<tr><td>Critérios de aprovação</td><td colspan="3">Aprovação em revisão gerencial.
Aprovação da Proposta de Especificação de Software pelo cliente.</td></tr>
</table>

Tabela 1 - Script da Ativação

Com base nas informações levantadas, são desenvolvidas as atividades do fluxo de Gestão. Deve-se fazer o levantamento das metas gerenciais, determinando limites de custos e prazos aceitáveis para o cliente, e as estimativas de custo e prazo da fase de Elaboração.

O resultado dessa fase é uma proposta que deverá ser aceita pelo cliente para que o projeto possa se iniciar realmente.

5.3.2 Elaboração

5.3.2.1 Levantamento dos Requisitos

Na iteração de Levantamento de Requisitos, o objetivo é detalhar os requisitos até um nível em que cliente, usuários e desenvolvedores possam chegar a um consenso. O Levantamento de Requisitos focaliza a visão que o cliente e usuários têm dos requisitos. Nesta iteração, os requisitos presentes na Proposta de Especificação são revisados e ampliados. Para auxiliar na captura dos requisitos, [Paula01] recomenda a utilização de oficinas estruturadas, contando com a participação de todas as partes interessadas. Podem ser feitos, também, protótipos rápidos e estudos de viabilidade, para auxiliar na definição dos requisitos de interface ou quanto à utilização ou não de determinada tecnologia. A Tabela 2 mostra o *script* para o Levantamento dos Requisitos.

O resultado desta iteração é a Especificação de Requisitos, mas ainda sem o Modelo de Análise. Os requisitos funcionais são expressos através de casos de uso. Os requisitos não funcionais são também definidos e descritos de forma textual. É feito um esboço das interfaces de usuários para se definir os requisitos de cada uma, sendo que detalhes de desenho devem ser evitados e postergados até a fase de Construção.

As atividades do fluxo de Criação de Conteúdo, nesta iteração, têm por objetivo formar o "conceito" do sítio. Essa etapa é muito importante para que o conteúdo final da aplicação esteja de acordo com a expectativa de seus usuários e com os objetivos do cliente. Com base nesse levantamento são definidos os requisitos de conteúdo e como será feita a manutenção desse conteúdo. Os requisitos gerais de conteúdo levantados junto ao cliente e aos usuários finais devem ser inseridos também na especificação de requisitos numa seção dedicada a eles. Aqueles requisitos que dizem respeito a cada uma das interfaces de usuário devem ser registrados junto ao esboço dessas interfaces. Quanto à manutenção de conteúdo, deve ser definido se ela será feita de forma automatizada ou não, se será adquirida uma ferramenta especial para esse fim ou, ainda, se será desenvolvido um módulo da aplicação através do qual a manutenção será feita. Neste último caso, os requisitos funcionais e as interfaces do novo módulo devem ser especificados e documentos.

As atividades do fluxo de usabilidade também podem ser resumidas como um levantamento e definição dos requisitos. O levantamento é feito através da análise dos usuários e tarefas. Com base nos dados levantados, são definidos os requisitos específicos de usabilidade que devem, também, ser inseridos na Especificação de Requisitos.

Ao final da iteração, todos os requisitos definidos devem ser lançados em um Cadastro de Requisitos. Esse cadastrado é criado para fins de rastreabilidade, pois servirá para amarrar os requisitos com os elementos derivados posteriormente nos demais fluxos.

Descrição	Levantamento detalhado das funções, interfaces e requisitos não funcionais desejados para o produto.		
Pré-requisitos	Ativação terminada.		
Insumos	PESw.		
Atividades	**Fluxo**	**Tarefas**	
	Requisitos	Levantamento completo dos requisitos. Detalhamento das interfaces. Detalhamento dos casos de uso. Detalhamento dos requisitos não funcionais.	
	Análise	Estudos de viabilidade (opcional).	
	Desenho	Estudos de viabilidade (opcional).	
	Implementação	Prototipagem dos requisitos (opcional).	
	Testes	Testes dos protótipos dos requisitos (opcional).	
	Criação de Conteúdo	Levantamento junto ao cliente. Levantamento junto aos usuários finais. Definição dos requisitos de conteúdo. Definição da estratégia de manutenção de conteúdo.	
	Usabilidade	Análise do usuário. Análise de tarefas. Definição dos requisitos de usabilidade.	
	Gestão	Cadastramento dos requisitos.	
Resultados	**Artefato**	**Sigla**	**Partes adicionadas**
	Modelo de Análise do Software	MASw	Visão de casos de uso.
	Especificação dos Requisitos do Software	ERSw	Corpo.
	Cadastro de Requisitos do Software	CRSw	Interfaces, casos de uso, requisitos não funcionais, requisitos de conteúdo e requisitos de usabilidade.
Critérios de aprovação	Aprovação em revisão gerencial.		

Tabela 2 - Script do Levantamento dos Requisitos

5.3.2.2 Análise dos Requisitos

A iteração de Análise dos Requisitos focaliza a visão que os desenvolvedores têm dos requisitos. Ao final da análise, a Especificação de Requisitos estará completa. O script para a Análise dos Requisitos é mostrado na Tabela 3.

O principal artefato a ser produzido nesta iteração é o Modelo de Análise. Segundo [Paula01], este modelo utiliza a notação orientada a objetos para descrever os conceitos do domínio da aplicação que sejam relevantes para o entendimento dos requisitos. O Modelo de Análise é criado em duas etapas. Na primeira, é criado o modelo lógico dos dados através da identificação das classes, seus atributos e os relacionamentos entre as classes. Na segunda, serão definidas as operações de cada classe, como é feita a comunicação entre elas. Estas operações

serão utilizadas na definição das realizações dos casos de uso. A descrição das realizações dos casos de uso irá validar os próprios casos de uso e as classes. Muitas vezes, são identificados problemas nos requisitos funcionais levantados, que devem, então, ser modificados e corrigidos.

Em paralelo a estas atividades, a equipe envolvida com o conteúdo estará em sua fase criativa, usando como base para sua inspiração os requisitos e o conceito geral levantados na iteração anterior. Os resultados desse processo criativo serão esboços para a arte gráfica das interfaces de usuário. De acordo com experiências vividas em projetos reais, é recomendado que o artista crie de antemão algumas variações no desenho[1] gráfico das páginas, para que o cliente possa optar por uma ou outra linha. Isso agilizaria o processo de aprovação do desenho gráfico, diminuindo os pedidos de alterações e a necessidade de muitas reuniões com o cliente. Durante a criação dos esboços, podem ser identificados problemas nos requisitos de conteúdo, que devem ser, então, corrigidos. Ao final destas atividades, deve ser produzida a primeira parte do Documento do Conteúdo do Software, contendo os esboços produzidos e suas variações.

A atividade principal do fluxo de Usabilidade, nesta iteração, é a de Planejamento das Avaliações de Usabilidade. Com base nos requisitos de usabilidade, levantados na iteração anterior, é possível fazer uma análise de custo/benefício para determinar o número e tipo de avaliações que serão feitas e, ainda, em quais fases elas serão necessárias. Deve ser criado um cronograma para as avaliações, determinando-se quando elas irão ocorrer e o prazo para que seus resultados sejam comunicados. O planejamento ainda deve incluir a alocação de recursos, a atribuição de responsabilidades e estimativas de custo e prazo das avaliações.

Tendo em mãos a Especificação de Requisitos completa, os esboços das interfaces e a definição das atividades referentes à usabilidade, pode-se elaborar o Plano de Desenvolvimento, que traz um planejamento detalhado do restante do projeto. Além disto, pode-se planejar também as atividades que devem ser realizadas para a garantia da qualidade, resultando em um Plano da Qualidade.

Ao final desta iteração, artefatos muito importantes terão sido produzidos, artefatos que constituem a base para o restante do projeto. A partir desse ponto, cliente e fornecedor estarão assumindo grandes riscos e compromissos. Por esses motivos, o PRAXIS recomenda diferentes procedimentos de controle, antes de se passar para a próxima fase. Os artefatos devem ser revisados quanto à parte técnica (revisão técnica) e quanto à conformidade com o que é determinado pelo processo (auditoria da qualidade). Deve, ainda, ser feita uma revisão gerencial, onde é verificado se os desenvolvedores concordam com os compromissos a serem assumidos com o cliente e é feito um balanço da iteração. Após as revisões internas, os artefatos devem ser submetidos ao cliente. Este os analisará e decidirá pela continuidade ou não do projeto.

[1] *Design*.

<table>
<tr><td>Descrição</td><td colspan="4">Modelagem conceitual dos elementos relevantes do domínio do problema e uso deste modelo para validação dos requisitos e planejamento detalhado da fase de Construção.</td></tr>
<tr><td>Pré-requisitos</td><td colspan="4">Levantamento dos requisitos terminado.</td></tr>
<tr><td rowspan="2">Insumos</td><td colspan="2">Antigos</td><td colspan="2">Novos</td></tr>
<tr><td colspan="2">PESw.</td><td colspan="2">ERSw - corpo; MASw - visão de casos de uso.</td></tr>
<tr><td rowspan="10">Atividades</td><td>Fluxo</td><td colspan="3">Tarefas</td></tr>
<tr><td>Requisitos</td><td colspan="3">Revisão e modificação dos requisitos (se necessário).</td></tr>
<tr><td>Análise</td><td colspan="3">Identificação das classes.
Identificação dos atributos e relacionamentos.
Realização dos casos de uso.
Revisão e iteração.</td></tr>
<tr><td>Desenho</td><td colspan="3">Estudos de viabilidade (opcional).
Desenho arquitetônico.</td></tr>
<tr><td>Implementação</td><td colspan="3">Prototipagem dos requisitos (opcional).</td></tr>
<tr><td>Testes</td><td colspan="3">Testes dos protótipos dos requisitos (opcional).</td></tr>
<tr><td>Criação de Conteúdo</td><td colspan="3">Processo criativo do desenho das interfaces.
Revisão e modificação dos requisitos de conteúdo (se necessário).</td></tr>
<tr><td>Usabilidade</td><td colspan="3">Revisão e modificação dos requisitos de usabilidade (se necessário).
Planejamento das avaliações de usabilidade.</td></tr>
<tr><td>Gestão</td><td colspan="3">Cadastramento dos itens de análise no cadastro de requisitos.
Planejamento do desenvolvimento.
Planejamento da qualidade.</td></tr>
<tr><td rowspan="9">Resultados</td><td colspan="2">Artefato</td><td>Sigla</td><td>Partes adicionadas</td></tr>
<tr><td colspan="2">Modelo de Análise do Software</td><td>MASw</td><td>Visão lógica.</td></tr>
<tr><td colspan="2">Especificação dos Requisitos do Software</td><td>ERSw</td><td>Anexo – listagem do modelo de análise.</td></tr>
<tr><td colspan="2">Cadastro de Requisitos do Software</td><td>CRSw</td><td>Classes.</td></tr>
<tr><td colspan="2">Documento do Conteúdo do Software</td><td>DCSw</td><td>Parte do desenho gráfico das interfaces de usuário.</td></tr>
<tr><td colspan="2">Memória de Cálculo do Projeto do Software</td><td>MCPSw</td><td>Total.</td></tr>
<tr><td colspan="2">Modelo de Planejamento do Projeto do Software</td><td>MPPSw</td><td>Total.</td></tr>
<tr><td colspan="2">Plano de Desenvolvimento do Software</td><td>PDSw</td><td>Total.</td></tr>
<tr><td colspan="2">Plano da Qualidade do Software</td><td>PQSw</td><td>Total (menos detalhes de implementação).</td></tr>
<tr><td>Critérios de aprovação</td><td colspan="4">Aprovação em revisão técnica.
Aprovação em auditoria da qualidade.
Aprovação em revisão gerencial.
Aprovação da Especificação dos Requisitos do Software, do Plano de</td></tr>
</table>

	Desenvolvimento do Software e do Desenho gráfico das Interfaces de Usuário pelo cliente.

Tabela 3 - Script da Análise dos Requisitos

5.3.3 Construção

5.3.3.1 Desenho Inicial

Segundo [Paula01], o Desenho Inicial é a iteração mais importante da Construção pois nela será produzida a arquitetura do sistema, serão definidas as soluções tecnológicas mais adequadas para satisfazer os requisitos e as questões técnicas importantes para a implementação. O *script* para o Desenho Inicial é mostrado na Tabela 4.

No PRAXIS, recomenda-se que o primeiro esboço do desenho arquitetônico seja feito durante a iteração de Análise dos Requisitos, pois ele é geralmente necessário para tomar decisões fundamentais de arquitetura e tecnologia, que influenciam bastante o custo e prazo da fase de Construção. A iteração de Desenho Inicial aprofunda o desenho arquitetônico, definindo de forma precisa os subsistemas que comporão o produto. Nesta atividade, o produto é dividido em camadas, pacotes lógicos e subsistemas, são identificadas tecnologias adequadas para a implementação e componentes já existentes que podem ser reutilizados. Seguem essa atividade a de desenho dos subsistemas e a de desenho dos dados persistentes. O desenho dos subsistemas define aspectos importantes de cada um e como é feita a interface entre eles. O desenho dos dados persistentes tem como objetivo definir a melhor solução para o acoplamento entre o desenho do sistema, que é orientado a objetos, e a arquitetura de sistemas de armazenamento de dados persistentes, como bancos de dados relacionais.

Durante estas atividades, as classes já definidas na fase de análise vão sendo refinadas, mais detalhadas, transformando-se em classes de desenho. Com as classes de desenho em mãos e partindo dos casos de uso de análise, são elaborados os casos de uso de desenho. O nível de detalhamento dos casos de uso e de suas realizações é maior do que na análise, ficando bem mais próximo do que será feito na implementação.

O desenho detalhado das interfaces de usuário também é feito nesta iteração. Para identificar o mais cedo possível problemas na usabilidade das interfaces e sua adequação com os requisitos de usabilidade, podem ser criados protótipos das interfaces. Estes protótipos podem, então, ser submetidos a avaliações de usabilidade. De acordo com os resultados das avaliações, o desenho das interfaces pode ser aperfeiçoado. Evita-se, assim, a necessidade de grandes mudanças no futuro, quando as interfaces já estiverem implementadas.

Em paralelo a todas as atividades já explicadas, a equipe responsável pelo conteúdo deve redigir a parte textual deste conteúdo, tendo sempre em mente os requisitos levantados anteriormente. Quando o conteúdo da aplicação é muito específico ou necessita de um maior nível de aprofundamento, a fase de redação deve ser precedida por uma de pesquisa, na qual podem ser consultados especialistas ou outras fontes de informação. Nesta iteração, deve ser feita também a captura de todo o material multimídia necessário para a produção do conteúdo não textual do produto. Esta captura compreende a gravação de sons e vídeo, a tomada de fotos,

digitalização de imagens, busca de imagens em bancos de imagens e a compra das mesmas (se for o caso). O objetivo das atividades do fluxo de Criação de Conteúdo, nesta iteração, é obter o material a ser trabalhado, editado e formatado na iteração seguinte.

<table>
<tr><td>Descrição</td><td colspan="4">Definição interna e externa dos componentes de um produto de software, a nível suficiente para decidir as principais questões de arquitetura e tecnologia, e para permitir o planejamento detalhado das atividades de implementação.</td></tr>
<tr><td>Pré-requisitos</td><td colspan="4">Elaboração terminada.</td></tr>
<tr><td rowspan="2">Insumos</td><td colspan="2">Antigos</td><td colspan="2">Novos</td></tr>
<tr><td colspan="2">MASw; ERSw; CRSw</td><td colspan="2">MCPSw; MPPSw; PDSw; PQSw; DCSw.</td></tr>
<tr><td rowspan="10">Atividades</td><td>Fluxo</td><td colspan="3">Tarefas</td></tr>
<tr><td>Requisitos</td><td colspan="3">Revisão e modificação dos requisitos (se necessário).</td></tr>
<tr><td>Análise</td><td colspan="3">Revisão e modificação do modelo de análise (se necessário).</td></tr>
<tr><td>Desenho</td><td colspan="3">Desenho dos subsistemas.
Desenho do acesso a dados persistentes.
Desenho das interfaces de usuário.
Elaboração dos casos de uso de desenho.
Desenho das liberações.</td></tr>
<tr><td>Implementação</td><td colspan="3">Prototipagem das interfaces de usuário (opcional).
Prototipagem de problemas de desenho (opcional).</td></tr>
<tr><td>Testes</td><td colspan="3">Planejamento e desenho dos testes de aceitação.</td></tr>
<tr><td>Criação de Conteúdo</td><td colspan="3">Redação dos textos.
Captura do material.</td></tr>
<tr><td>Usabilidade</td><td colspan="3">Execução das avaliações de usabilidade com as interfaces de usuário.
Comunicação dos resultados das avaliações.</td></tr>
<tr><td>Gestão</td><td colspan="3">Cadastramento dos itens de teste no cadastro de requisitos.
Cadastramento dos itens de desenho no cadastro de requisitos.
Cadastramento dos itens de conteúdo no cadastro de requisitos.
Cadastramento dos resultados das avaliações de usabilidade.
Planejamento detalhado das liberações.
Atualização dos planos de desenvolvimento e da qualidade.</td></tr>
<tr><td rowspan="8">Resultados</td><td colspan="2">Artefato</td><td>Sigla</td><td>Partes</td></tr>
<tr><td colspan="2">Insumos</td><td></td><td>Eventuais modificações e detalhes adicionais.</td></tr>
<tr><td colspan="2">Modelo de Desenho do Software</td><td>MDSw</td><td>Todas as visões, em descrição de alto nível.</td></tr>
<tr><td colspan="2">Descrição do Desenho do Software</td><td>DDSw</td><td>Partes 1 a 4, com colocação no anexo das partes já produzidas do MDSw</td></tr>
<tr><td colspan="2">Documento do Conteúdo do Software</td><td>DCSw</td><td>Parte dos textos elaborados para as interfaces.</td></tr>
<tr><td colspan="2">Material de Conteúdo de Multimídia do Software</td><td>MCMSw</td><td>Arquivos de imagens, sons, vídeo.</td></tr>
<tr><td colspan="2">Relatórios das Avaliações de Usabilidade do Software</td><td>RAUSw</td><td>Resultados das avaliações de usabilidade (opcional).</td></tr>
<tr><td colspan="2">Descrição dos Testes do Software</td><td>DTSw</td><td>Planos e especificações dos testes de aceitação.</td></tr>
</table>

	Bateria de Testes de Regressão do Software	BTRSw	Scripts para automação dos testes de aceitação (opcional).
Critérios de aprovação	Aprovação em revisão técnica. Aprovação do desenho das interfaces de usuário pelos usuários chaves. Aprovação em auditoria da qualidade. Aprovação em revisão gerencial.		

Tabela 4 - Script do Desenho Inicial

Com o produto dividido em subsistemas bem definidos e com as realizações dos casos de uso de desenho, pode-se, então, planejar as liberações. Neste planejamento, deve-se definir quantas liberações serão feitas e quais subsistemas serão implementados por cada uma delas.

Ao final do Desenho Inicial, os artefatos devem ser submetidos a uma revisão técnica. Deve ser também realizada uma auditoria da qualidade, para verificar a conformidade com o processo, e uma revisão gerencial. O desenho das interfaces de usuário deverá também ser aprovado por usuários chave, antes de se passar para a próxima etapa.

5.3.3.2 Liberação

Para cada liberação definida no Desenho Inicial, deve ser realizado um conjunto de atividades que irão, basicamente, partindo do desenho, gerar uma porção executável do produto que poderá ser submetida à avaliação dos usuários. A Tabela 5 traz o *script* de uma liberação típica.

A primeira atividade a ser realizada é o desenho detalhado dos componentes da liberação. Esta atividade irá resolver aspectos ainda faltantes, como assinatura e visibilidade das operações das classes, implementação dos relacionamentos, tratamento de erros e exceção, algoritmos e estruturas de dados mais adequados. Em seqüência, os testes de integração e unidade são planejados e o desenho detalhado é submetido a uma inspeção.

Passa-se, então, à codificação, onde o desenho detalhado será convertido em código nas linguagens de implementação escolhidas. Nessa etapa, o conteúdo disponibilizado para a liberação (pela equipe de criação de conteúdo) deve ser integrado ao código. O código deve ser inspecionado e, se estiver de acordo com o desenho e os padrões de codificação, será compilado. São realizados, então, os testes de unidade. Feitos os testes e corrigidos os problemas, o código da liberação será integrado aos componentes já produzidos e, se for o caso, aos componentes externos. O conjunto é submetido aos testes de integração.

As atividades referentes à criação de conteúdo têm como objetivo, nesta iteração, processar e disponibilizar, já no formato correto, os itens de conteúdo que dizem respeito a essa liberação. Os textos devem ser revisados e formatados. Os sons e vídeos devem ser editados, as animações produzidas. As imagens devem ser tratadas para que tenham a qualidade desejada, o formato desejado. As imagens que fazem parte da estrutura das páginas devem ser recortadas. Todo esse conteúdo deve ser submetido a uma revisão e, então, disponibilizado para a equipe de codificação. É importante que a equipe responsável pelo conteúdo esteja sempre adiantada em relação à codificação, para que ao final do desenho detalhado o conteúdo da liberação já esteja disponível para a equipe de codificação.

<table>
<tr><td>Descrição</td><td colspan="4">Implementação de um subconjunto de funções do produto que será avaliado pelos usuários.</td></tr>
<tr><td>Pré-requisitos</td><td colspan="4">Desenho Inicial terminado.
Liberação anterior terminada e avaliada pelos usuários.</td></tr>
<tr><td rowspan="2">Insumos</td><td colspan="3">Antigos</td><td>Novos</td></tr>
<tr><td colspan="3">MASw; ERSw; CRSw; MCPSw; MPPSw; PDSw; PQSw; MDSw; DDSw; DTSw; BTRSw; DCSw; RAUSw; MCMSw.</td><td>CFSw (liberação anterior); CESw (liberação anterior); RTSw (liberação anterior).</td></tr>
<tr><td rowspan="10">Atividades</td><td>Fluxo</td><td colspan="3">Tarefas</td></tr>
<tr><td>Requisitos</td><td colspan="3">Revisão e modificação dos requisitos (se necessário).</td></tr>
<tr><td>Análise</td><td colspan="3">Revisão e modificação do modelo de análise (se necessário).</td></tr>
<tr><td>Desenho</td><td colspan="3">Revisão e modificação do modelo de desenho (se necessário).</td></tr>
<tr><td>Implementação</td><td colspan="3">Desenho detalhado dos componentes desta liberação.
Codificação dos componentes desta liberação.
Compilação dos componentes desta liberação.</td></tr>
<tr><td>Testes</td><td colspan="3">Planejamento e desenho dos testes de integração da liberação.
Planejamento e desenho dos testes de unidade da liberação.
Realização dos testes de unidade da liberação.
Realização dos testes de integração da liberação.</td></tr>
<tr><td>Criação de Conteúdo</td><td colspan="3">Revisão e formatação dos textos da liberação.
Edição e pós-processamento do material da liberação.
Revisão do conteúdo da liberação.
Disponibilização do conteúdo da liberação (para os desenvolvedores).</td></tr>
<tr><td>Usabilidade</td><td colspan="3">Execução das avaliações de usabilidade.
Comunicação dos resultados das avaliações.</td></tr>
<tr><td>Gestão</td><td colspan="3">Revisão e modificação dos planos de desenvolvimento e da qualidade (se necessário).</td></tr>
<tr><td rowspan="8">Resultados</td><td colspan="2">Artefato</td><td>Sigla</td><td>Partes</td></tr>
<tr><td colspan="2">Insumos</td><td></td><td>Eventuais modificações e detalhes adicionais.</td></tr>
<tr><td colspan="2">Documento do Conteúdo do Software</td><td>DCSw</td><td>Parte dos textos revisada e com formatação correta especificada. Elementos multimídia de cada interface relacionados, cores especificadas, arquivos correspondentes e caminhos especificados.</td></tr>
<tr><td colspan="2">Material de Conteúdo de Multimídia do Software</td><td>MCMSw</td><td>Arquivos já editados e no formato correto a ser utilizado.</td></tr>
<tr><td colspan="2">Relatórios das Avaliações de Usabilidade do Software</td><td>RAUSw</td><td>Resultados das avaliações de usabilidade.</td></tr>
<tr><td colspan="2">Relatórios dos Testes do Software</td><td>RTSw</td><td>Relatórios dos testes de unidade e de integração da liberação (opcional).</td></tr>
<tr><td colspan="2">Códigos Fontes do Software</td><td>CFSw</td><td>Unidades da liberação.
Estruturas provisórias de teste.</td></tr>
<tr><td colspan="2">Códigos Executáveis do Software</td><td>CESw</td><td>Unidades da liberação.
Estruturas provisórias de teste.</td></tr>
</table>

Critérios de aprovação	Aprovação em inspeção do desenho detalhado e código da liberação. Aprovação do conteúdo pelo cliente. Aprovação da liberação pelos usuários chaves. Aprovação em auditoria da qualidade. Aprovação em revisão gerencial.

Tabela 5 - Script da Liberação n

Uma vez executável, e liberação deve ser submetida aos usuários finais para avaliação. Neste ponto, é pertinente realizar uma avaliação de usabilidade, para identificar problemas que ainda tenham passado despercebidos pelas avaliações anteriores.

Para se passar para a próxima iteração, a liberação deve ser aprovada pelos usuários chave, deve ser aprovada em uma auditoria da qualidade e deve, também, ser realizada uma revisão gerencial.

5.3.3.3 Testes Alfa

Nesta iteração, são realizados os testes de aceitação no ambiente dos desenvolvedores. É recomendável que estes testes sejam executados por uma equipe independente da equipe de desenvolvimento. Os problemas encontrados pelos testes devem ser corrigidos e rastreados no desenho e nos requisitos. O *script* para os Testes Alfa é apresentado na Tabela 6.

A fase de Transição deve ser, então, planejada e deve ser produzido o material de apoio a ela. O principal item deste material é o Manual do Usuário. No caso de aplicações Web, o mais comum é se disponibilizar no próprio sítio a documentação necessária para que o usuário esclareça suas dúvidas.

Para garantir a adequação com os requisitos de usabilidade, pode-se, ainda, realizar avaliações de usabilidade nesta iteração. Apesar de não ser obrigatório, é interessante avaliar como o comportamento e dificuldades encontradas pelo usuário, tendo em vista o sistema completo.

O procedimento de controle previsto para o final dos Testes Alfa é uma auditoria da qualidade. Após a auditoria, o sistema pode ser entregue ao cliente para que seja implantado nas instalações da fase de Transição. Nos casos em que o próprio fornecedor for hospedar a aplicação Web, ele próprio fará esta implantação.

<table>
<tr><td>Descrição</td><td colspan="4">Realização dos testes de aceitação, no ambiente dos desenvolvedores, juntamente com elaboração da documentação de usuário e possíveis planos de Transição.</td></tr>
<tr><td>Pré-requisitos</td><td colspan="4">Última liberação terminada e avaliada pelos usuários.</td></tr>
<tr><td rowspan="2">Insumos</td><td colspan="2">Antigos</td><td colspan="2">Novos</td></tr>
<tr><td colspan="2">MASw; ERSw; CRSw; MCPSw; MPPSw; PDSw; PQSw; MDSw; DDSw; DTSw; BTRSw.</td><td colspan="2">CFSw última (liberação); CESw (última liberação); RTSw (última liberação); DCSw; RAUSw; MCMSw.</td></tr>
<tr><td rowspan="9">Atividades</td><td>Fluxo</td><td colspan="3">Tarefas</td></tr>
<tr><td>Requisitos</td><td colspan="3">Revisão e modificação dos requisitos (se necessário).</td></tr>
<tr><td>Análise</td><td colspan="3">Revisão e modificação do modelo de análise (se necessário).</td></tr>
<tr><td>Desenho</td><td colspan="3">Revisão e modificação do desenho de alto nível (se necessário).</td></tr>
<tr><td>Implementação</td><td colspan="3">Revisão e modificação do desenho detalhado e código (se necessário).
Produção da documentação de usuário.</td></tr>
<tr><td>Testes</td><td colspan="3">Realização dos testes alfa (aceitação no ambiente dos desenvolvedores).</td></tr>
<tr><td>Criação de Conteúdo</td><td colspan="3">Revisão e modificação do conteúdo (se necessário).</td></tr>
<tr><td>Usabilidade</td><td colspan="3">Execução das avaliações de usabilidade (opcional).
Comunicação dos resultados das avaliações (opcional).</td></tr>
<tr><td>Gestão</td><td colspan="3">Planejamento detalhado da Transição.
Atualização dos planos de desenvolvimento e da qualidade.</td></tr>
<tr><td rowspan="4">Resultados</td><td colspan="2">Artefato</td><td>Sigla</td><td>Partes</td></tr>
<tr><td colspan="2">Insumos</td><td></td><td>Eventuais modificações e detalhes adicionais.</td></tr>
<tr><td colspan="2">Relatórios das Avaliações de Usabilidade do Software</td><td>RAUSw</td><td>Resultados das avaliações de usabilidade (opcional).</td></tr>
<tr><td colspan="2">Relatórios dos Testes do Software</td><td>RTSw</td><td>Relatórios dos testes alfa.</td></tr>
<tr><td colspan="2">Manual do Usuário do Software</td><td>MUSw</td><td></td></tr>
<tr><td>Critérios de aprovação</td><td colspan="4">Aprovação em auditoria da qualidade.
Aprovação em revisão gerencial.
Aprovação da entrega do produto pelo cliente.</td></tr>
</table>

Tabela 6 - Script dos Testes Alfa

5.3.4 Transição

5.3.4.1 Testes Beta

Nesta iteração, são realizados os testes de aceitação nas instalações dos usuários. A documentação de usuário também é testada. Podem ser identificados problemas relacionados com o funcionamento em instalações reais, que devem ser corrigidos. A Tabela 7 traz o *script* para os Testes Beta.

<table>
<tr><td>Descrição</td><td colspan="4">Realização dos testes de aceitação, no ambiente dos usuários.</td></tr>
<tr><td>Pré-requisitos</td><td colspan="4">Construção terminada.
Aceitação da instalação do produto pelo cliente.</td></tr>
<tr><td rowspan="2">Insumos</td><td colspan="2">Antigos</td><td colspan="2">Novos</td></tr>
<tr><td colspan="2">MASw; ERSw; CRSw; MCPSw; MPPSw; PDSw; PQSw; MDSw; DDSw; DTSw; BTRSw; DCSw; MCMSw.</td><td colspan="2">RTSw (testes alfa); MUSw; RAUSw (testes alfa).</td></tr>
<tr><td rowspan="9">Atividades</td><td>Fluxo</td><td colspan="3">Tarefas</td></tr>
<tr><td>Requisitos</td><td colspan="3">Revisão e modificação dos requisitos (se necessário).</td></tr>
<tr><td>Análise</td><td colspan="3">Revisão e modificação do modelo de análise (se necessário).</td></tr>
<tr><td>Desenho</td><td colspan="3">Revisão e modificação do desenho de alto nível (se necessário).</td></tr>
<tr><td>Implementação</td><td colspan="3">Revisão e modificação do desenho detalhado e código (se necessário).
Revisão e modificação da documentação de usuário (se necessário).</td></tr>
<tr><td>Testes</td><td colspan="3">Realização dos testes beta (aceitação no ambiente dos usuários).</td></tr>
<tr><td>Criação de Conteúdo</td><td colspan="3">Revisão e modificação do conteúdo (se necessário).</td></tr>
<tr><td>Usabilidade</td><td colspan="3">Estudos de continuidade (opcional).</td></tr>
<tr><td>Gestão</td><td colspan="3">Revisão e modificação dos planos de desenvolvimento e da qualidade (se necessário).</td></tr>
<tr><td rowspan="3">Resultados</td><td colspan="2">Artefato</td><td>Sigla</td><td>Partes</td></tr>
<tr><td colspan="2">Insumos</td><td></td><td>Eventuais modificações e detalhes adicionais.</td></tr>
<tr><td colspan="2">Relatórios dos Estudos de Continuidade de Usabilidade do Software</td><td>RECUSw</td><td>Estudos de continuidade durante os testes beta (opcional).</td></tr>
<tr><td colspan="2">Relatórios dos Testes do Software</td><td>RTSw</td><td>Relatórios dos testes beta.</td></tr>
<tr><td>Critérios de aprovação</td><td colspan="4">Aprovação em auditoria da qualidade.
Aprovação em revisão gerencial.
Aprovação dos testes beta pelo cliente.</td></tr>
</table>

Tabela 7 - Script dos Testes Beta

Se ainda forem encontrados problemas relacionados aos requisitos, ao desenho ou ao conteúdo, estes devem ser corrigidos e os respectivos documentos atualizados.

Como atividade opcional do fluxo de Usabilidade, podem ser realizados estudos de continuidade durante os testes beta. Embora esta atividade seja mais importante depois que o produto está sendo realmente utilizado pelos usuários, pode-se ir colhendo informações desde os testes que, quando analisadas, apontem possíveis melhorias a serem feitas numa próxima versão.

Os procedimentos de controle para esta iteração incluem uma auditoria da qualidade, uma revisão gerencial e a aprovação dos testes beta pelo cliente.

5.3.4.2 Operação Piloto

Segundo [Paula01], a Operação Piloto representa um estado de "liberdade vigiada" do produto. Os problemas encontrados nesta iteração pelos usuários são tratados pelo processo de manutenção. Assim os serviços de manutenção e suporte ao usuário são também colocados em prova. O *script* para a Operação Piloto é mostrado na Tabela 8.

Nesta iteração e durante todo o período que o sistema estiver em produção, podem ser realizados estudos de continuidade (atividade do fluxo de Usabilidade). O objetivo principal destes estudos é obter informações de usabilidade para a próxima versão do produto.

Durante a Operação Piloto, é feito um balanço final do projeto onde são levantados os principais problemas de processo encontrados e possíveis melhorias. Este balanço é documentado através do Relatório Final do Projeto.

<table>
<tr><td>Descrição</td><td colspan="4">Operação experimental do produto em instalação piloto do cliente, com a resolução de eventuais problemas através de processo de manutenção.</td></tr>
<tr><td>Pré-requisitos</td><td colspan="4">Testes beta terminados e aprovados pelo cliente.</td></tr>
<tr><td rowspan="2">Insumos</td><td colspan="2">Antigos</td><td colspan="2">Novos</td></tr>
<tr><td colspan="2">MASw; ERSw; CRSw; MCPSw; MPPSw; PDSw; PQSw; MDSw; DDSw; DTSw; BTRSw; DCSw; RAUSw; MCMSw; MUSw.</td><td colspan="2">RTSw (testes beta); RECUSw.</td></tr>
<tr><td rowspan="9">Atividades</td><td>Fluxo</td><td colspan="3">Tarefas</td></tr>
<tr><td>Requisitos</td><td colspan="3">Revisão e modificação dos requisitos (se necessário).</td></tr>
<tr><td>Análise</td><td colspan="3">Revisão e modificação do modelo de análise (se necessário).</td></tr>
<tr><td>Desenho</td><td colspan="3">Revisão e modificação do desenho de alto nível (se necessário).</td></tr>
<tr><td>Implementação</td><td colspan="3">Revisão e modificação do desenho detalhado e código (se necessário).
Revisão e modificação da documentação de usuário (se necessário).</td></tr>
<tr><td>Testes</td><td colspan="3">Revisão e modificação da documentação de testes (se necessário).</td></tr>
<tr><td>Criação de Conteúdo</td><td colspan="3">Revisão e modificação do conteúdo (se necessário).</td></tr>
<tr><td>Usabilidade</td><td colspan="3">Estudos de Continuidade (opcional).</td></tr>
<tr><td>Gestão</td><td colspan="3">Balanço final do projeto.
Produção do Relatório Final do Projeto.</td></tr>
<tr><td rowspan="3">Resultados</td><td colspan="2">Artefato</td><td>Sigla</td><td>Partes</td></tr>
<tr><td colspan="2">Insumos</td><td></td><td>Eventuais modificações e detalhes adicionais.</td></tr>
<tr><td colspan="2">Relatórios dos Estudos de Continuidade de Usabilidade do Software</td><td>RECUSw</td><td>Estudos de continuidade durante a operação piloto (opcional).</td></tr>
<tr><td colspan="2">Relatório Final de Projeto de Software</td><td>RFPSw</td><td></td></tr>
<tr><td>Critérios de aprovação</td><td colspan="4">Aprovação em Auditoria da Qualidade.
Aprovação em Revisão Gerencial.
Aceitação final do produto pelo cliente.</td></tr>
</table>

Tabela 8 - Script da Operação Piloto

Capítulo 6

Estudo de Caso

6.1 Introdução

Para verificar se o processo criado é realmente factível, foi desenvolvida uma aplicação simples, mas que põe à prova os aspectos mais importantes que o processo se dispõe a melhorar ou dar suporte. O objetivo principal do estudo de caso realizado é verificar se as atividades criadas para os novos fluxos foram distribuídas de forma correta e em uma seqüência lógica entre as diferentes fases do processo. Outro objetivo é verificar se os artefatos propostos cumprem sua finalidade de documentar as decisões tomadas e auxiliar na comunicação entre os diferentes profissionais.

Para dar continuidade ao exemplo elaborado para o PRAXIS, decidiu-se criar uma versão para a Web do sistema Merci[1]. O Merci é um sistema que provê o apoio informatizado ao controle de vendas, de compras, de fornecedores e de estoque da mercearia Pereira & Pereira Comercial Ltda. A versão para a Web do Merci será chamada de e-Merci e será descrita na seção seguinte.

6.2 Desenvolvimento

6.2.1 Concepção

Na fase de concepção, foi definido o escopo do sistema e foram levantados seus requisitos principais. Foi realizado, também, um levantamento preliminar dos requisitos de conteúdo e de usabilidade para a aplicação.

Para demonstrar como o tempo de disponibilização do produto no mercado pode ser diminuído utilizando-se o artifício de dividi-lo em versões incrementais, foram definidas três versões para o e-Merci. Na primeira, serão disponibilizadas as funcionalidades básicas para que o cliente da mercearia consiga efetuar uma compra pela Internet e ter esta compra entregue em casa. A segunda e terceira versões irão acrescentar funcionalidades também importantes que

[1] Exemplo utilizado em [Paula01].

agregam valor ao produto, facilitando ainda mais a venda pela Internet.

Para cada versão, foi elaborada, então, uma Proposta de Especificação de Software, seguindo o padrão do PRAXIS e acrescentando somente uma seção para documentar os requisitos de conteúdo levantados.

As missões identificadas para cada uma das versões são mostradas na Tabela 9.

O produto e-Merci 1.0 visa disponibilizar para os atuais clientes da mercearia Pereira & Pereira Ltda a possibilidade de comprar via Internet e atrair novos clientes.
O e-Merci 1.1 visa ampliar as facilidades dadas aos clientes através do e-Merci 1.0. Nesta versão, será dada uma maior flexibilidade na forma de pagamento e os clientes poderão pagar suas contas encomendadas através do sítio utilizando cartões de crédito.
O e-Merci 1.2 visa auxiliar no relacionamento e conhecimento dos clientes do sítio. Esta versão agregará às outras o cadastro de clientes.

Tabela 9 - Missão de cada uma das versões do e-Merci

As listas de funções determinadas para o e-Merci 1.0, e-Merci 1.1 e e-Merci 1.2 são mostradas nas tabelas Tabela 10, Tabela 11, Tabela 12, respectivamente.

Número de ordem	Nome da função	Necessidades	Benefícios
1	Operação de venda pela Internet	Realizar venda pela Internet	Maior comodidade e agilidade para os clientes da mercearia
2	Identificação do Usuário (cliente da mercearia)	Identificar nome do cliente e endereço de entrega	Possibilidade de recebimento das compras em casa
3	Emissão da lista de compras	Emitir uma lista para o cliente com as mercadorias encomendadas	Maior facilidade para o cliente da mercearia na conferência do que foi recebido

Tabela 10 – Lista de Funções para o e-Merci 1.0

Número de ordem	Nome da função	Necessidades	Benefícios
1	Operação de venda com cartão de crédito	Possibilitar aos clientes pagar suas compras através de cartão de crédito	Maior facilidade no pagamento para os clientes

Tabela 11 – Lista de Funções para o e-Merci 1.1

Número de ordem	Nome da função	Necessidades	Benefícios
1	Cadastro de Clientes	Armazenar os dados dos clientes do sítio	Facilidade no conhecimento do número de clientes e suas características Facilidade na Operação de Venda para o cliente, que não precisa preencher os dados novamente a cada compra

Tabela 12 – Lista de Funções para o e-Merci 1.2

Os requisitos de conteúdo, definidos neste levantamento preliminar, foram:

- No projeto gráfico das páginas devem prevalecer as cores do logotipo da mercearia Pereira & Pereira: azul e laranja.
- Os textos das páginas não devem ser formais, devem se aproximar da linguagem oral o quanto for possível, mas sem erros.

Os requisitos de usabilidade, definidos nesta fase, foram:

- A carga das páginas não deve ultrapassar 15 segundos.
- Os clientes devem conseguir realizar suas compras sem dificuldades em, pelo menos, 60% dos casos.

Tanto os requisitos de conteúdo quanto os requisitos de usabilidade são válidos para as três versões previstas do produto.

Foram ainda identificados outros aspectos importantes a serem considerados na fase de Elaboração para cada uma das versões. Estes aspectos estão listados na Tabela 13.

O produto fará a interface do sistema já utilizado na mercearia, o Merci, com o cliente via Web. O produto trata somente das vendas a serem feitas via Internet, sendo que a atualização da base de dados e demais aspectos continuam sendo de responsabilidade do Merci. Nesta versão 1.0, o pagamento será feito na entrega das compras. Este procedimento não será informatizado.
A versão 1.1 agrega valor à versão 1.0 do e-Merci, incluindo, como forma de pagamento possível, o cartão de crédito. O cliente informará o número de seu cartão de crédito e um funcionário da mercearia processará a cobrança antes da entrega das compras. Deverá ser adotada alguma técnica para garantir a segurança do número do cartão do cliente.
A versão 1.2 agrega valor à versão 1.1 do e-Merci. Ela terá todas as funcionalidades da anterior e ainda o cadastro de clientes. O cadastro deverá armazenar informações básicas do cliente, mas deve ser feito de forma a, no futuro, poderem ser incluídas mais informações.

Tabela 13 – Outros aspectos importantes para cada uma das versões do e-Merci

As metas gerenciais identificadas foram que a versão 1.0 deveria ser desenvolvida em, no máximo, uma semana, com um custo de até R$ 500,00; a versão 1.1 deveria ser desenvolvida em, no máximo, quatro dias, com um custo de até R$ 400,00; a versão 1.2 deveria ser desenvolvida

em, no máximo, uma semana, com um custo de até R$ 500,00.

Foram identificadas as atividades a serem realizadas para a especificação de cada versão e o tempo necessário para se realizá-las. O prazo e custo final para uma versão só poderão ser identificados ao final da Elaboração desta versão.

6.2.2 Elaboração

Após a Concepção, iniciou-se a fase de Elaboração para a versão 1.0 do produto. Nesta fase, os requisitos funcionais, as interfaces de usuário, os requisitos não funcionais, os requisitos de usabilidade e de conteúdo foram definidos e documentados de forma completa. Foi realizado, também, o planejamento da fase de Construção da versão 1.0.

Para o e-Merci 1.0 foram elaborados todos os artefatos previstos para esta fase do WebPraxis, sendo eles: MASw, ERSw, CRSw, DCSw, MCPSw, MPPSw e PDSw. O PQSw foi incluído no PDSw, por ser bastante simples no caso deste projeto. Serão apresentados a seguir, os aspectos mais importantes dos artefatos ERSw, DCSw e PDSw.

6.2.2.1 Especificação de Requisitos do Software

A Especificação de Requisitos do Software para o e-Merci 1.0 foi elaborada levando-se em consideração os pontos importantes para uma aplicação web destacados na seção 5.2.1.1 deste documento. Alguns aspectos da ERSw são mostrados a seguir.

A missão do e-Merci 1.0 permaneceu a mesma identificada na fase de Concepção:

O e-Merci 1.0 visa atrair novos clientes para a mercearia Pereira & Pereira Ltda e disponibilizar, para os atuais, a possibilidade de comprar via Internet.

Foram identificados os limites para a versão 1.0, ou seja, aspectos que não serão tratados nesta versão:

O e-Merci 1.0 não oferecerá recursos adicionais de segurança, além dos embutidos nos clientes e servidores da Web.
O e-Merci 1.0 não possibilitará o pagamento on-line e nem disponibilizará a funcionalidade de impressão de boleto bancário.

Os benefícios proporcionados pelo produto foram listados e classificados de acordo com seu valor para o cliente. No caso da versão 1.0, todos os benefícios foram considerados essenciais. A Tabela 14 traz a lista destes benefícios.

Número de ordem	Benefício	Valor para o Cliente
1	Maior comodidade e agilidade para os usuários em sua tarefa de fazer compras	Essencial
2	Facilidade na conferência das compras encomendadas	Essencial
3	Possibilidade de entrega das compras na casa do usuário	Essencial

Tabela 14 - Benefícios do produto e-Merci 1.0

O diagrama apresentado na Figura 14, traz as principais funções identificadas para o produto e os atores envolvidos em cada uma delas.

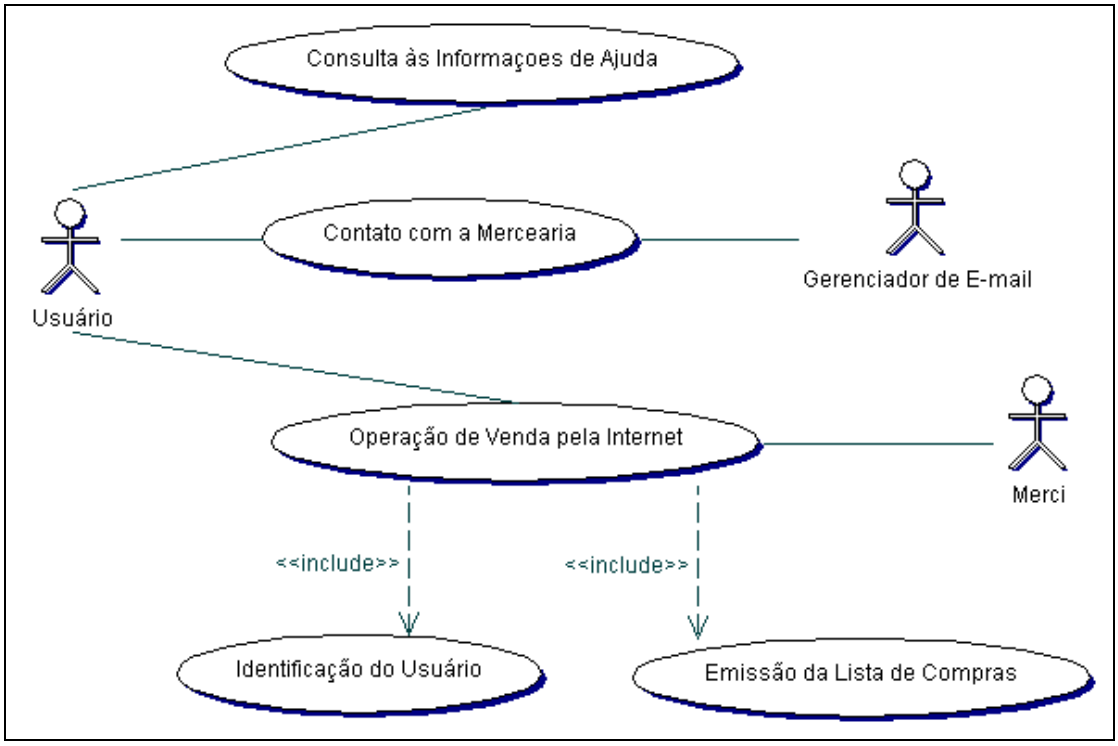

Figura 14 – Diagrama de contexto do e-Merci 1.0

As interfaces de usuário elaboradas para o e-Merci 1.0 estão descritas na Tabela 15. Não foram identificadas, para esta versão, interfaces de hardware ou de comunicação que diferem das tradicionalmente disponibilizadas pelo sistema operacional. Como interface de software foram identificadas as conexões com o Merci e com o Gerenciador de correio eletrônico.

O e-Merci 1.0 possuirá somente um modo de operação, o modo de venda, onde todas as suas funções estarão disponíveis a seus usuários.

Número de ordem	Nome	Ator	Caso de uso	Descrição
1	Tela de Abertura	Usuário	-	Tela de abertura do sistema, por onde o usuário tem acesso às outras funcionalidades.
2	Tela de Escolha das Mercadorias	Usuário	Operação de Venda pela Internet	Tela onde o usuário escolhe as mercadorias e indica a quantidade desejada.
3	Tela de Efetivação da Compra	Usuário	Identificação do Usuário	Tela onde o usuário informa seu nome, endereço e horário para entrega das compras e finaliza suas compras.
4	Tela de Emissão da Lista de Compras	Usuário	Emissão da Lista de Compras	Tela onde o usuário pode visualizar, de forma resumida, todos os itens encomendados e imprimir esta lista.
5	Tela de Ajuda à Operação de Venda	Usuário	Consulta às Informações de Ajuda	Tela onde o usuário pode esclarecer alguma dúvida sobre como realizar sua compra.
6	Tela de Contato com a Mercearia	Usuário	Contato com a Mercearia	Tela onde o usuário pode entrar em contato com a mercearia através de correio eletrônico.

Tabela 15 – Interfaces de usuário do e-Merci 1.0

As funções identificadas para esta versão do e-Merci, assim como a caracterização de seus usuários, se encontram listadas na Tabela 16 e na Tabela 17, respectivamente.

Número de ordem	Caso de uso	Descrição
1	Operação de Venda pela Internet	Disponibiliza os produtos cadastrados no Merci para os usuários escolherem e guarda quais foram escolhidos. O usuário pode alterar essa escolha ou confirmá-la.
2	Consulta às Informações de Ajuda	Exibe as informações de ajuda ao usuário na realização de suas compras.
3	Contato com a Mercearia	Disponibiliza um formulário simples para o usuário entrar em contato com a Mercearia, através de correio eletrônico, para esclarecimento de dúvidas, envio de sugestões, etc.
4	Emissão da Lista de Compras	Exibe, de forma resumida, uma lista das mercadorias escolhidas pelo usuário e possibilita sua impressão.
5	Identificação do Usuário	Disponibiliza um formulário simples para o usuário informar nome, telefone, endereço e horário para a entrega de sua compra.

Tabela 16 – Funções do e-Merci 1.0

Número de ordem	Atores	Permissão de acesso	Freqüência de uso	Nível educacional	Proficiência na aplicação	Proficiência em Informática
1	Usuário	Todas as funções do produto.	Esporádica	1º Grau, no mínimo	Mínima	Windows 98

Tabela 17 – Características dos usuários do e-Merci 1.0

Foram identificados alguns requisitos que não seriam tratados nessa versão. Tais requisitos foram incluídos no documento como requisitos adiados, que poderão ser tratados nas próximas versões do produto. A Tabela 18 apresenta os requisitos adiados.

Número de ordem	Referência ao requisito	Detalhes
1	Cadastro de Clientes da Mercearia	Armazenagem das informações sobre os usuários, para que ele não precise informar sempre que for fazer uma compra.
2	Possibilidade de pagamento das compras feitas através do sistema com cartão de crédito	Incluir a possibilidade de pagamento com cartão, permitindo que o usuário informe o número de seu cartão e protegendo o tráfego dessa informação na rede.
3	Cadastro de Ofertas	Permitir aos funcionários da mercearia cadastrar ofertas que apareceriam em destaque na Mercearia Virtual.

Tabela 18 – Requisitos adiados, não tratados na versão 1.0

Foram levantados os requisitos para as interfaces gráficas de usuário. A seguir, é mostrado um exemplo do detalhamento de uma das interfaces de usuário[1].

O exemplo é para a interface de usuário Tela de Efetivação de Compra. O leiaute sugerido para esta interface é mostrado na Figura 15. Deve ser lembrado que o leiaute sugerido na especificação de requisitos não é o leiaute definitivo da interface de usuário. O leiaute definitivo será obtido na fase de desenho. No caso do WebPraxis, o leiaute definitivo depende também do processo criativo dos profissionais envolvidos no fluxo de criação de conteúdo. Como o estudo de caso foi realizado num curto período de tempo, optou-se por já elaborar nesta fase o leiaute definitivo das interfaces, mas não é esta a regra.

[1] Todas as interfaces de usuário listadas na Tabela 15 foram detalhadas, mas o que pretende-se aqui é mostrar somente um exemplo.

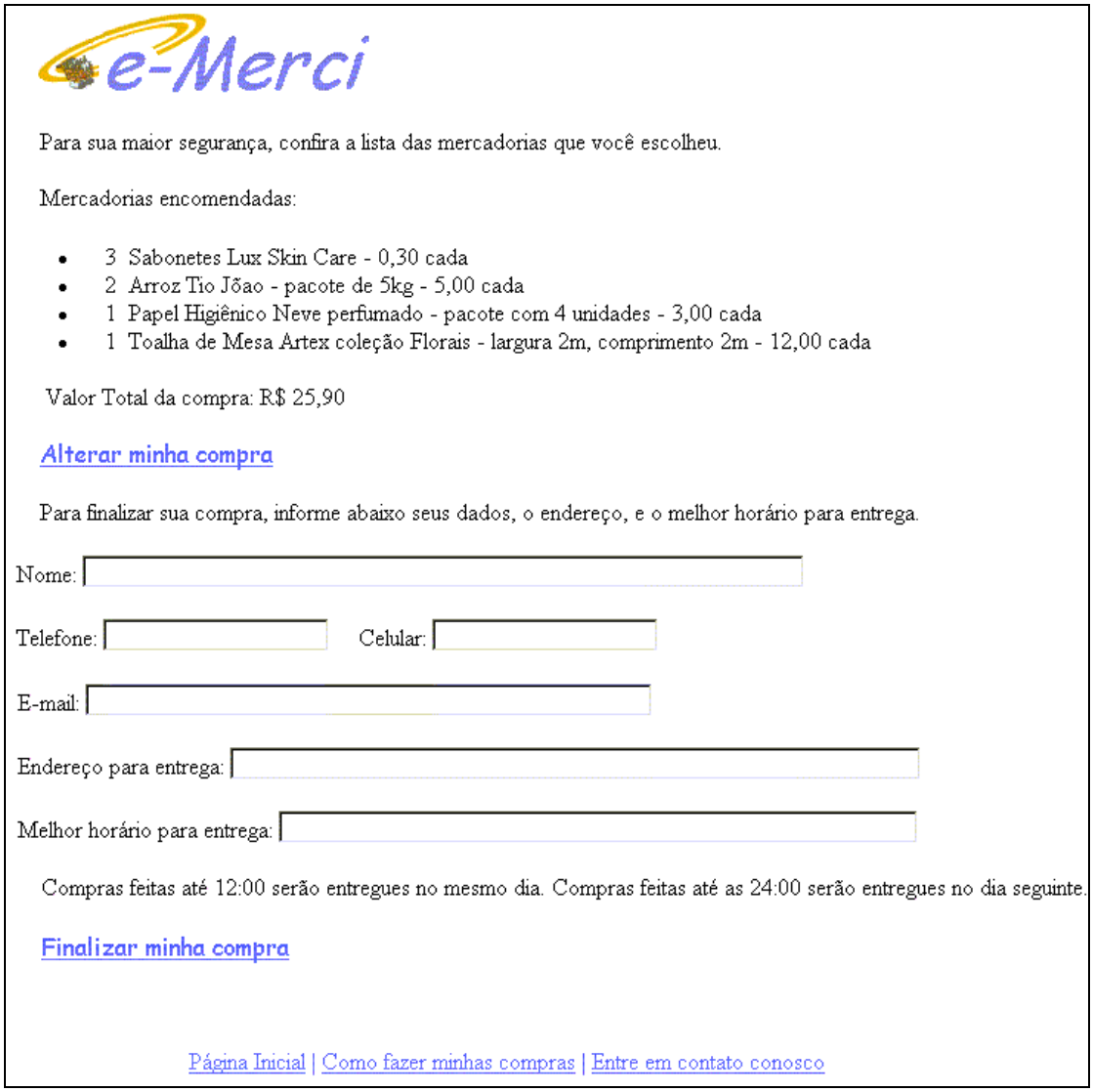

Figura 15 - Tela de Efetivação de Compra – Leiaute sugerido

Os relacionamentos da Tela de Efetivação de Compra com outras interfaces são listados na Tabela 19.

A hiperligação Página Inicial aciona a Tela de Abertura.
A hiperligação Como fazer suas compras aciona a Tela de Ajuda.
A hiperligação Entre em contato conosco aciona a Tela de Contato com a Mercearia.

Tabela 19 – Tela de Efetivação de Compra – Relacionamentos com outras interfaces

Os campos e comandos levantados para esta interface são listados e descritos na Tabela 20 e na Tabela 21, respectivamente.

Número	Nome	Valores válidos	Formato	Tipo	Restrições
1	Mercadorias encomendadas / Quantidade desejada	Maior ou igual a zero.	Até 4 algarismos.	Inteiro	Preenchido pelo e-Merci / Não alterável.
2	Mercadorias encomendadas / Descrição	Não vazio.	Até 100 caracteres.	Texto	Preenchido pelo e-Merci / Não alterável.
3	Mercadorias encomendadas / Preço	Não vazio.	4 casas e 2 casas decimais.	Moeda	Preenchido pelo e-Merci / Não alterável.
4	Valor Total da compra	Não vazio.	5 casas e 2 casas decimais.	Moeda	Calculado como o somatório de (Preço da Mercadoria escolhida x Quantidade Desejada) para cada Mercadoria escolhida / Não alterável.
5	Nome	Não vazio.	Até 60 caracteres.	Texto	Obrigatório / Alterável.
6	Telefone	Não vazio.	Até 11 caracteres.	Texto	Obrigatório / Alterável.
7	Celular	-	Até 11 caracteres.	Texto	Opcional / Alterável.
8	E-mail	-	Até 50 caracteres.	Texto	Opcional / Alterável.
9	Endereço para a entrega	Não vazio.	Até 120 caracteres.	Texto	Obrigatório / Alterável.
10	Melhor horário para a entrega	Não vazio.	Até 50 caracteres.	Texto	Obrigatório / Alterável.

Tabela 20 – Tela de Efetivação de Compra – Campos

Número	Nome	Ação	Restrições
1	Alterar minha compra	Aciona a Tela de Escolha das Mercadorias, em seu estado anterior (com as mercadorias escolhidas pelo usuário)	Sempre habilitado.
2	Finalizar minha compra	Aciona a Tela de Emissão da Lista de Compras e envia para a Mercearia uma mensagem eletrônica contendo todos os dados da encomenda do usuário, com o arquivo de entrada para o Merci anexado.	Habilitado depois que o usuário preencher os campos obrigatórios.

Tabela 21 – Tela de Efetivação de Compra – Comandos

Após levantar e detalhar os requisitos de cada uma das interfaces, tem-se o detalhamento dos casos de uso. O diagrama de contexto apresentado anteriormente é dividido em diagramas mais

específicos e mais detalhados. Um dos diagramas no qual foi dividido o diagrama de contexto do e-Merci 1.0 é apresentado na Figura 16. Este diagrama representa o caso de uso principal do e-Merci 1.0: Operação de Venda pela Internet.

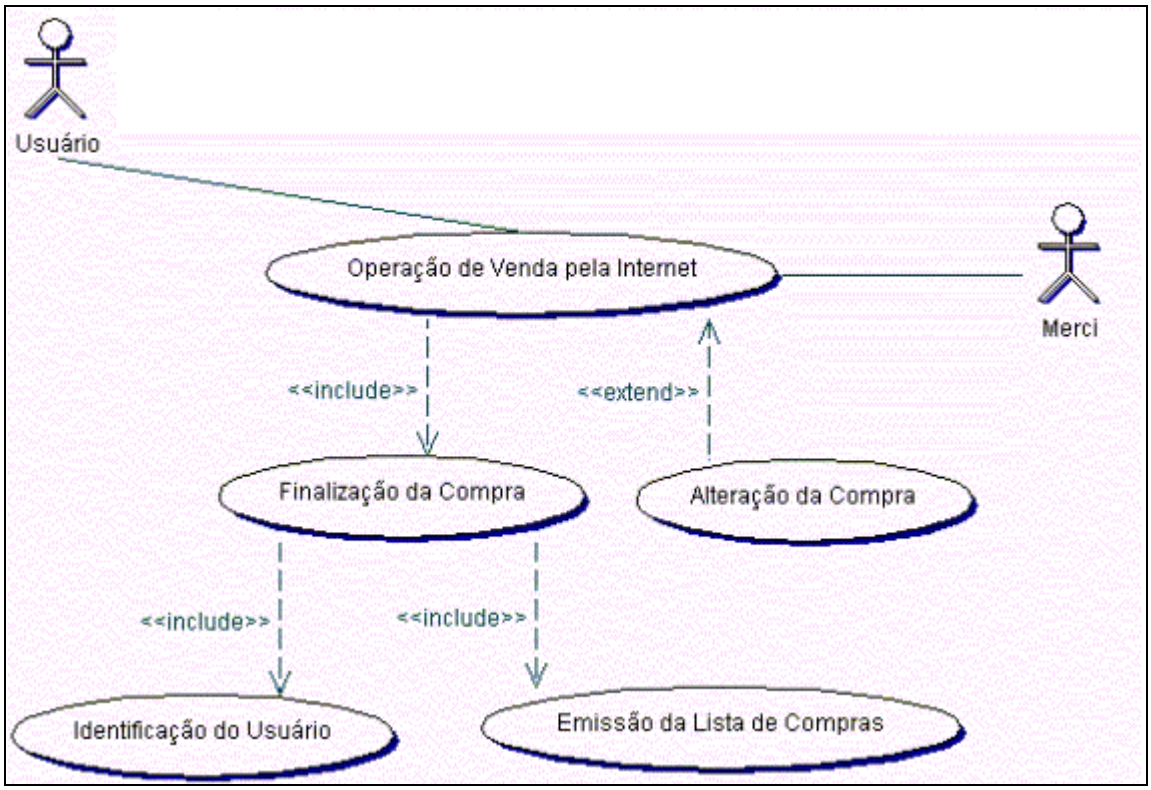

Figura 16 – Caso de uso Operação de Venda pela Internet

Além de sua representação através de diagramas, cada caso de uso identificado para o produto deve ser descrito de forma textual para esclarecer aspectos que não são expressos nos diagramas. A Tabela 22 apresenta um exemplo de descrição de caso de uso.

Caso de uso Operação de Venda pela Internet
Precondições
O Usuário deve ter solicitado o início de suas compras.
Fluxo principal
O e-Merci exibe a lista das mercadorias disponíveis.
O Usuário escolhe as mercadorias e indica as quantidades desejadas.
Se o Usuário pedir a totalização de sua compra,
Se houver alguma mercadoria selecionada, o e-Merci totaliza a compra e exibe a lista do que foi encomendado pelo usuário.
Se o Usuário pedir para alterar sua compra, o e-Merci aciona o fluxo alternativo Alteração da Compra.
Caso contrário, o e-Merci aciona o subfluxo Finalização da Compra.
Se o Usuário escolher ir para outra tela,

O e-Merci abre a tela escolhida.

Subfluxos

Subfluxo Finalização da Compra

O e-Merci aciona o subfluxo Identificação do Usuário.

Se o Usuário escolher finalizar suas compras,

O e-Merci gera o arquivo de interface com o Merci.

O e-Merci gera e envia a mensagem eletrônica para a Mercearia com os dados da compra e o arquivo de interface com o Merci anexado.

O e-Merci aciona o subfluxo Emissão da Lista de Compras.

Se o Usuário escolher ir para outra tela,

O e-Merci abre a tela escolhida.

Subfluxo Identificação do Usuário

O e-Merci exibe um formulário para o usuário preencher seus dados.

O Usuário preenche seus dados com nome, telefone, celular, e-mail, endereço para a entrega e horário para a entrega.

Subfluxo Emissão da Lista de Compras

O e-Merci exibe os dados do cliente e da compra.

Se o Usuário escolher ir para outra tela,

O e-Merci abre a tela escolhida.

Fluxos alternativos

Fluxo alternativo Alteração da Compra

O e-Merci exibe a lista das mercadorias disponíveis, com as mercadorias e as quantidades já escolhidas anteriormente pelo Usuário.

O Usuário escolhe novas mercadorias, ou retira mercadorias do pedido, e indica as quantidades desejadas.

O e-Merci volta ao fluxo principal.

Tabela 22 – Descrição do caso de uso Operação de Venda pela Internet

A descrição dos casos de uso e seus diagramas se referem aos requisitos funcionais do produto. Durante a fase de elaboração, são identificados, ainda, os requisitos não funcionais. Muitos destes requisitos não funcionais são extremamente importantes para as aplicações web, como descrito na seção 5.2.1.1 deste documento. Devem ser incluídos como requisitos não funcionais os requisitos de conteúdo, que irão servir de guia para as atividades do fluxo de criação de conteúdo.

Os requisitos não funcionais identificados para o e-Merci versão 1.0, são apresentados na Tabela 23.

Requisitos de desempenho

A carga das páginas montadas dinamicamente não deve ultrapassar 15 segundos, sendo 5 segundos o tempo ideal de carga.

Requisitos de segurança

O e-Merci 1.0 não oferecerá recursos adicionais de segurança, além dos embutidos nos clientes e servidores da Web.

Requisitos de portabilidade

O e-Merci deve ser perfeitamente visualizado com o *Microsoft Internet Explorer 5.0* ou superior, ou o *Netscape 4.01* ou superior, em qualquer plataforma para as quais estes softwares dêem suporte.

Requisitos da qualidade

Requisito da qualidade Apreensibilidade

Os clientes que iniciarem uma compra devem conseguir completá-la em, pelo menos, 60% dos casos, sem recorrer à página de ajuda.

Requisitos de conteúdo

Requisitos de conteúdo Cores Predominantes

No projeto gráfico das páginas, devem prevalecer as cores azul e laranja.

Requisitos de conteúdo Estilo do Projeto Gráfico

O projeto gráfico das páginas deve ser simples, limpo, de forma a não distrair o usuário de sua tarefa primordial: realizar sua compra.

Requisito de conteúdo Estilo dos Textos

Os textos apresentados nas páginas devem estar corretos, mas devem procurar se aproximar da linguagem oral, facilitando seu entendimento pelos usuários.

Requisito de conteúdo Clareza dos Textos

Os textos que indicam as ações a serem tomadas pelos usuários para realizar a compra devem ser claros e precisos (botões de ação).

Tabela 23 – Requisitos não funcionais do e-Merci 1.0

Para completar o documento de especificação dos requisitos, são incluídos os diagramas do modelo de análise. Nos diagramas de classes, são apresentadas todas as classes identificadas durante a análise, divididas em classes de fronteira, classes de controle e classes de entidade. Nos diagramas de seqüência e/ou colaboração, são representadas as interações entre as classes identificadas para realizar os casos de uso.

Para o e-Merci 1.0 foram elaborados os diagramas de classes e os diagramas de seqüência para os casos de uso mais relevantes.

6.2.2.2 Documento do Conteúdo do Software

As atividades do fluxo de criação de conteúdo realizadas na fase de Elaboração são centradas em captar os requisitos de conteúdo, documentá-los e elaborar os esboços do desenho gráfico das interfaces de usuário. Os requisitos levantados para o conteúdo foram documentados como requisitos não funcionais na especificação de requisitos. Num primeiro momento, foi criado um artefato chamado Desenho Gráfico das Interfaces de Usuário (DGIUSw), para documentar os esboços das interfaces de usuário elaborados pela equipe de criação de conteúdo, na iteração de análise dos requisitos.

Após elaborar este artefato, identificou-se que ele poderia ser unido ao artefato Documento de Conteúdo Textual do Software (que seria produzido já na fase de Construção), e, ainda, que era preciso incluir aspectos referentes ao conteúdo de cada página que não estavam sendo documentados anteriormente. Sendo assim, os dois artefatos foram substituídos pelo DCSw (Documento do Conteúdo do Software). Este novo artefato começaria a ser feito na iteração de análise dos requisitos e continuaria na fase de construção. No novo documento, para cada página da aplicação é exibido seu desenho gráfico, são explicitados quais os recursos multimídia presentes, são listados os textos, é indicada a formatação dos textos, são especificadas as cores utilizadas (em RGB, por exemplo) e é também indicado o caminho onde os arquivos de conteúdo podem ser encontrados.

6.2.2.3 Plano de Desenvolvimento do Software

Para mensurar o tamanho do projeto a fim de elaborar o plano de desenvolvimento, foi realizado um cálculo de pontos de função levando-se em consideração o que havia sido especificado. O resultado encontrado foi de 36 pontos de função para o e-Merci versão 1.0.

Com base neste resultado, foi elaborado o plano de desenvolvimento. Os principais marcos definidos para o projeto são mostrados na Tabela 24.

Número de ordem	Nome da atividade	Data inicial prevista	Data final prevista
1	Desenho	20/3/2001	26/3/2001
2	Construção da Liberação Executável 1	26/3/2001	13/4/2001
3	Testes Alfa	13/4/2001	17/4/2001
4	Testes Beta	17/4/2001	18/4/2001
5	Operação Piloto	19/4/2001	19/4/2001

Tabela 24 – Principais marcos para o e-Merci 1.0

Os mecanismos de monitoração e controle estipulados são mostrados na Tabela 25.

Número de ordem	Tipo de revisão	Documentos revisados	Tópicos revisados	Marco associado à revisão	Data prevista
1	Revisão Técnica	Descrição do Desenho do Software	Todos	Fim do Desenho Inicial	26/3/2001
2	Revisão Técnica	Descrição dos Testes do Software	Descrição dos testes de aceitação	Fim do Desenho Inicial	26/3/2001
3	Inspeção	Descrição do Desenho do Software	Desenho detalhado	Antes da codificação da Liberação 1	30/3/2001
4	Inspeção	Códigos fontes	Todos	Antes dos testes da Liberação 1	10/04/2001

Tabela 25 – Mecanismos de monitoração e controle para o e-Merci 1.0

A Tabela 26 traz o cronograma final obtido para o projeto.

Etapa	Início	Fim	Duração (dias)
Desenvolvimento total	20/3/2001	19/4/2001	23,00
Construção	20/3/2001	17/4/2001	20,75
Desenho Inicial	20/3/2001	26/3/2001	4,75
Liberação 1	26/3/2001	13/4/2001	13,75
Testes Alfa	13/4/2001	17/4/2001	2,25
Transição	17/4/2001	19/4/2001	2,25
Testes Beta	17/4/2001	18/4/2001	1,25
Operação Piloto	19/4/2001	19/4/2001	1,00

Tabela 26 – Cronograma para o e-Merci 1.0

6.2.3 Construção

Após a Concepção, iniciou-se a fase de Construção da versão 1.0 do produto. Nesta fase, foi realizada somente uma parte das atividades da primeira iteração, o Desenho Inicial.

Foi elaborado um protótipo das interfaces de usuário com as telas criadas na fase de Elaboração. Devido ao pouco tempo, as telas não sofreram alterações no desenho. Esse protótipo foi utilizado para a realização de um teste simples de usabilidade com usuários reais. Os testes foram realizados de uma maneira informal mas, ainda assim, possibilitaram conclusões importantes a respeito das interfaces. Nielsen, em [Nielsen00], afirma que uma interface para a web deve utilizar o mínimo possível o recurso de barra de rolagem, pois o usuário na web dificilmente irá utilizar a barra e visualizar o que não aparece num primeiro momento na tela. Este aspecto ficou comprovado no teste realizado: na interface onde o usuário deveria finalizar sua compra, quando a lista de compras exibida era muito grande, a opção de finalizar a compra

ficava escondida num primeiro momento, aparecendo somente no final da página. Os usuários simplesmente não conseguiam finalizar a compra pois não descobriam onde estava esta opção.

Nesta iteração, foi definida a arquitetura do e-Merci 1.0. O padrão escolhido foi o de cliente espesso, visto que o projeto será implementado utilizando-se páginas ASP.

Quanto ao acesso aos dados persistentes, ficou definido que o e-Merci 1.0 não os acessa diretamente. Todo acesso aos dados persistentes é feito através do Merci.

Foram elaborados os textos para as páginas e para a página de ajuda. Estes textos foram inseridos no DCSw (Documento do Conteúdo do Software).

6.3 Conclusão

Mesmo realizando um estudo de caso bastante simples, este se mostrou importante para a identificação de pontos a serem melhorados ou trabalhados de forma diferente no WebPraxis.

Foi possível identificar que os artefatos criados para o fluxo de Criação de Conteúdo poderiam ser unificados em um único documento que começaria a ser feito na iteração de análise dos requisitos e continuaria na fase de construção. Esse novo documento, engloba os antigos DGIUSw (Desenho Gráfico das Interfaces de Usuário) e DCTSw (Documento de Conteúdo Textual do Software) e ainda outros aspectos que não estavam sendo documentados. No novo documento, DCSw (Documento do Conteúdo do Software), para cada página da aplicação são explicitados quais os recursos multimídia presentes, são listados os textos, é indicada a formatação dos textos, são especificadas as cores utilizadas (em RGB, por exemplo) e é também indicado o caminho onde os arquivos de conteúdo podem ser encontrados.

Um outro ponto explicitado, que deveria ser tratado de forma diferente, surgiu na parte de planejamento. Ao se contar os pontos de função para a aplicação e se tentar fazer a conversão de pontos de função em linhas de código, não é possível utilizar as tabelas já existentes para as várias linguagens de programação, pois numa mesma aplicação para a Web, são utilizadas várias linguagens diferentes. As estimativas de esforço devem ser feitas, então, com base em experiências anteriores levando-se em consideração o tamanho do sistema em pontos de função e não em linhas de código.

As atividades do fluxo de usabilidade são de extrema importância para uma aplicação web, fato este que nos levou a colocá-las num fluxo separado a ser realizado por pessoas especializadas nesta área. Esta importância foi confirmada na experiência do estudo de caso. Ao realizarmos um teste de usabilidade com os protótipos das interfaces, foram identificados dois problemas críticos, que interferiam diretamente na conclusão de uma compra pelos usuários. Se esses problemas não tivessem sido identificados para que pudessem ser corrigidos antes do produto ser disponibilizado para seu público, muito provavelmente não seria possível atender ao requisito de qualidade presente da Especificação de Requisitos. Mesmo num projeto tão simples, as atividades referentes à usabilidade se mostraram extremamente importantes, principalmente os testes. Um desenvolvedor nunca consegue perceber todos os problemas que podem surgir para seu usuário ao tentar utilizar o sistema. É preciso que os próprios usuários, ou um conjunto representante dos mesmos, indiquem os pontos problemáticos. Isto é feito através dos testes de

usabilidade.

No geral, o estudo de caso mostrou que o processo proposto é factível. A distribuição das novas atividades entre as fases se mostrou correta quanto à ordem temporal e lógica em que o produto vai sendo construído.

Capítulo 7

Conclusão e Trabalhos Futuros

7.1 Conclusão

Apesar do caráter extremamente dinâmico das aplicações Web, hoje, e do reduzido tempo para chegar ao mercado que essas aplicações têm, é extremamente importante que seu desenvolvimento seja feito dentro de um processo bem definido e organizado. Desta forma, estas aplicações poderão evoluir, crescer em complexidade e funcionalidades de uma forma organizada e sem gerar tanto retrabalho, como, a cada nova versão, ter que reconstruir toda a aplicação. A utilização de um processo para aplicações Web é tão importante quanto sua utilização no desenvolvimento de outros tipos de aplicações. Este processo deve ser adaptado para que não se torne simplesmente um procedimento burocrático e, assim, um *overhead* inaceitável na entrega do produto final. Para diminuir o tempo de disponibilização no mercado, o desenvolvimento de uma aplicação Web pode ser planejado em várias versões, de forma que a primeira versão seja colocada rapidamente no ar e as posteriores agreguem novas funcionalidades. Para que este tipo de planejamento seja possível, um processo bem definido e que favoreça o reuso se torna indispensável.

Ainda hoje, muitas empresas conseguem entregar uma aplicação sem ter um processo muito definido, pois o nível de complexidade das aplicações Web é relativamente baixo se comparado com o de outros paradigmas. Mas essa complexidade vem crescendo rapidamente e só conseguirão se adaptar e continuar produzindo com qualidade e no tempo necessário aqueles que adotarem um processo bem definido e adequado para seus objetivos.

Como já foi colocado, o processo gerado por esse trabalho é uma especialização do Praxis para desenvolvimento de aplicações Web. Ainda assim, ele pode ser visto como um processo genérico para o desenvolvimento de aplicações web, cobrindo todos os aspectos importantes para esse tipo de projeto. Para ser empregado em alguma organização real, ele pode ser ainda adaptado de acordo com os interesses e objetivos da organização, podendo ser simplificado em algumas partes ou mais aprofundado em outras. Tudo irá depender das aplicações a serem produzidas e o que elas exigem. Dessa forma, mesmo dentro de uma mesma organização, o processo tem que ser adaptado de projeto para projeto. Nestes casos, não será necessária uma adaptação formal como a apresentada neste trabalho, mas será necessário precisar quais as atividades do processo serão

utilizadas e quais podem ser reduzidas ou descartadas.

Nosso objetivo inicial foi alcançado, pois conseguimos traçar as linhas gerais a serem seguidas num desenvolvimento para a Web, tentando resolver possíveis problemas e chamando a atenção dos analistas para os aspectos importantes a serem analisados. O próximo passo, agora, é aplicar o processo em projetos reais e começar sua evolução.

7.2 Trabalhos Futuros

Como continuação deste trabalho, seria interessante adequar os documentos e gabaritos que fazem parte do material do PRAXIS para o WebPraxis para que a documentação produzida pelos dois processos seja padronizada.

Outro trabalho interessante seria adequar a ferramenta *Praxis Mentor*, resultado da dissertação de mestrado da aluna Júnia G. Carvalho [Carvalho01], para dar suporte também ao WebPraxis.

Como dito anteriormente, é necessário, agora, aplicar o WebPraxis em projetos reais, pois sempre surgem novos entraves e possíveis melhorias. De nada adianta criar um processo e engavetá-lo. É preciso colocá-lo em prática e, na medida em que for sendo utilizado, incluir novas idéias, dar suporte a novas tecnologias e melhorar os pontos fracos identificados.

Uma das oportunidades já identificadas para se aplicar o WebPraxis é utilizá-lo na disciplina de Sistemas Multimídia. O processo auxiliará os alunos no desenvolvimento de seus trabalhos finais, que são, na maioria das vezes, aplicativos voltados para a Web. Como dito anteriormente, essa experiência é de grande importância, pois possibilitará um constante aperfeiçoamento no processo, na medida em que forem sendo percebidos pontos não cobertos ou não completamente satisfatórios.

Apêndice A

Tabelas da Definição do WebPraxis

Atividades do Processo x Atributos do Produto

A tabela seguinte apresenta o relacionamento das atividades do processo com os atributos do produto, classificando-o em fraco, médio e forte. Os atributos do produto também foram classificados como sendo de prioridade baixa, média e alta.

Os valores apresentados na tabela foram obtidos atribuindo-se pesos 2, 4 e 10[1] tanto para os atributos do produto quanto para o grau de relacionamento de uma atividade com o atributo. Os pesos foram multiplicados e depois somados para cada linha da tabela. O total apresentado na última coluna refere-se à atividade apresentada naquela linha, mostrando sua prioridade relativa em relação às outras.

Obs.: estão marcadas com asterisco (*) aquelas atividades que não estão presentes no PRAXIS e que foram incluídas no WebPRAXIS.

[1] Pesos sugeridos por Humphrey em [Humphrey95].

Relacionamento entre os atributos do produto e as atividades do processo											
Prioridade	**Alta (x 10)**			**Média (x 4)**					**Baixa (x 2)**		**Total**
Atributos do Produto⇒ ⇓Atividades do Processo	Suprir as necessidades do cliente	Não ultrapassar limites de custo e prazo	Gerar o mínimo de manutenções corretivas	Dar o suporte necessário à segurança dos dados	Ser de fácil utilização	Ter um design gráfico atrativo, que cative os usuários	Ter um bom desempenho	Dar o suporte necessário à portabilidade	Favorecer o reuso	Facilitar manutenção e expansão	
Determinação de um contexto	Médio 40	Médio 40	Fraco 20	Fraco 8	Fraco 8	Médio 16	Fraco 8	Fraco 8			148
Definição do escopo do produto	Forte 100	Forte 100	Fraco 20	Forte 40	Médio 16	Médio 16	Médio 16	Médio 16			324
Definição dos requisitos	Forte 100	Forte 100	Médio 40	Forte 40	Forte 40	Forte 40	Médio 16	Forte 40			416
Detalhamento dos requisitos de interface	Médio 40		Médio 40			Médio 16					96
Detalhamento dos casos de uso	Médio 40		Médio 40								80
Detalhamento dos requisitos não funcionais	Médio 40		Médio 40	Forte 40			Forte 40	Forte 40			200
Classificação dos requisitos	Médio 40	Forte 100	Fraco 20	Médio 16				Médio 16			192
Revisão dos requisitos	Forte 100		Forte 100	Médio 16			Médio 16	Médio 16			248
Aprovação da Especificação dos Requisitos do Software pelo cliente	Forte 100	Médio 40	Médio 40	Médio 16			Fraco 8	Médio 16			220
Identificação das classes			Médio 40						Médio 8	Forte 20	68
Organização das classes			Médio 40						Médio 8	Forte 20	68
Identificação dos relacionamentos			Médio 40						Médio 8	Médio 8	56
Identificação dos atributos e herança			Médio 40						Médio 8	Forte 20	68
Realização dos casos de uso (análise)			Médio 40						Médio 8	Médio 8	56

Revisão da análise	Médio 40		Forte 100						Médio 8	Médio 8	156
Desenho arquitetônico			Médio 40	Médio 16			Médio 16	Médio 16	Forte 20	Forte 20	128
Desenho dos dados persistentes			Médio 40	Médio 16			Médio 16	Médio 16	Forte 20	Forte 20	128
Detalhamento das classes de desenho			Médio 40	Médio 16			Médio 16	Médio 16	Médio 8	Médio 8	104
Desenho das interfaces de usuário			Médio 40				Forte 40	Forte 40	Médio 8	Médio 8	136
Realização dos casos de uso (desenho)			Médio 40				Médio 16	Médio 16	Médio 8	Médio 8	88
Desenho das liberações			Médio 40	Médio 16			Médio 16	Médio 16	Médio 8	Médio 8	104
Revisão do desenho	Médio 40		Forte 100	Médio 16			Médio 16	Médio 16	Médio 8	Médio 8	204
Planejamento dos testes	Médio 40		Forte 100	Médio 16			Forte 40	Forte 40	Fraco 4	Fraco 4	244
Desenho dos testes	Médio 40		Forte 100	Forte 40			Forte 40	Forte 40	Fraco 4	Fraco 4	268
Implementação dos testes	Médio 40		Forte 100	Forte 40			Forte 40	Forte 40	Fraco 4	Fraco 4	268
Execução dos testes	Médio 40		Forte 100	Forte 40			Forte 40	Forte 40	Fraco 4	Fraco 4	268
Verificação do término dos testes	Fraco 20		Médio 40	Médio 16			Médio 16	Médio 16	Fraco 4	Fraco 4	116
Balanço final dos testes	Fraco 20		Fraco 20	Fraco 8			Fraco 8	Fraco 8	Fraco 4	Fraco 4	72
Desenho detalhado	Médio 40		Médio 40	Médio 16			Médio 16	Médio 16	Médio 8	Médio 8	144
Revisão do desenho detalhado	Médio 40		Forte 100	Médio 16			Médio 16	Médio 16	Médio 8	Médio 8	204
Codificação	Médio 40		Médio 40	Médio 16			Médio 16	Médio 16	Médio 8	Médio 8	144
Revisão do código	Médio 40		Forte 100	Médio 16			Médio 16	Médio 16	Fraco 4	Fraco 4	196
Compilação	Médio 40		Médio 40	Fraco 8			Médio 16	Médio 16	Fraco 4	Fraco 4	128

Testes de unidade	Médio 40		Forte 100	Médio 16			Médio 16	Médio 16	Fraco 4	Fraco 4	180
Integração	Médio 40		Médio 40	Fraco 8			Fraco 8	Fraco 8	Fraco 4	Fraco 4	112
Análise do usuário	Forte 100				Forte 40						140
Análise das tarefas	Forte 100				Forte 40						140
Definição dos requisitos de usabilidade	Forte 100				Forte 40						140
Planejamento da usabilidade					Forte 40						40
Planejamento das avaliações de usabilidade					Forte 40						40
Execução das avaliações de usabilidade					Forte 40						40
Comunicação dos resultados das avaliações					Forte 40						40
Levantamento junto ao cliente (equipe marketing)	Forte 100					Forte 40					140
Levantamento junto aos usuários finais	Forte 100					Forte 40					140
Definição dos requisitos específicos do design gráfico	Forte 100					Forte 40					140
Criação do design						Forte 40					140
Alterações do design gráfico a pedido do cliente	Forte 100					Médio 16					116
Aprovação do design gráfico pelo cliente	Forte 100					Médio 16					116
Recorte das imagens						Médio 16					16

Montagem do esqueleto das interfaces de usuário						Médio 16					16
Levantamento das metas gerenciais		Médio 40									40
Estimativas da fase de elaboração		Forte 100									100
Elaboração da proposta de especificação		Médio 40									40
Planejamento do desenho		Forte 100	Fraco 20								120
Aprovação do Plano de desenvolvimento pelo cliente	Médio 40		Fraco 20								60
Planejamento da qualidade		Médio 40	Fraco 20								60
Planejamento detalhado das liberações		Forte 100	Fraco 20								120
Revisão e modificação dos planos de desenvolvimento e da qualidade		Forte 100	Fraco 20								120
Planejamento detalhado da transição		Forte 100	Fraco 20								120

Tabela 27 – Atividades do Processo x Atributos do Produto

Atividades do Processo x Objetivos do Processo

A tabela seguinte apresenta o relacionamento das atividades do processo com seus objetivos, classificando-o em fraco, médio e forte. Os atributos do produto também foram classificados como sendo de prioridade baixa, média e alta.

Os valores numéricos apresentados na tabela foram obtidos atribuindo-se pesos 2, 4 e 10[1] tanto para os objetivos do processo quanto para o grau de relacionamento de uma atividade com o objetivo. Os pesos foram multiplicados e depois somados para cada linha da tabela. O total apresentado na última coluna refere-se à atividade apresentada naquela linha, mostrando sua prioridade relativa em relação às outras. Foram listados também os totais obtidos na tabela anterior e a soma dos dois, para chegar à prioridade relativa das atividades considerando os dois aspectos, atributos do produto e objetivos do processo.

Obs.: estão marcadas com asterisco (*) aquelas atividades que não estão presentes no PRAXIS e que foram incluídas no WebPRAXIS.

[1] Pesos sugeridos por Humphrey em [Humphrey95].

Relacionamento entre os objetivos e as atividades do processo													
Prioridade	**Alta (x 10)**				**Média (x 4)**				**Baixa (x 2)**		**TOTAL Tab II**	**TOTAL Tab I**	**GERAL**
Objetivos do Processo⇒ ⇓Atividades do Processo	Auxiliar no cumprimento dos custos e prazos estipulados	Minimizar o número de manutenções corretivas	Minimizar o custo dos testes	Minimizar o tempo que o produto leva para chegar ao mercado	Possibilitar estimativas de custo e prazo mais próximas à realidade	Otimizar a utilização dos recursos disponíveis	Auxiliar no conhecimento da força de trabalho da empresa	Aumentar a qualidade do software produzido	Favorecer o reuso	Gerar produtos de fácil manutenção e expansão			
Determinação de um contexto	Médio 40	Fraco 20	Fraco 20	Fraco 20	Fraco 8			Fraco 8			116	148	264
Definição do escopo do produto	Forte 100	Fraco 20	Fraco 20	Forte 100	Médio 16			Médio 16			272	324	596
Definição dos requisitos	Forte 100	Médio 40	Médio 40	Forte 100	Forte 40			Forte 40			360	416	776
Detalhamento dos requisitos de interface		Médio 40	Médio 40								80	96	176
Detalhamento dos casos de uso		Médio 40	Médio 40								80	80	160
Detalhamento dos requisitos não funcionais		Médio 40	Médio 40								80	200	280
Classificação dos requisitos	Forte 100	Fraco 20		Forte 100							220	192	412
Revisão dos requisitos		Forte 100	Forte 100	Médio 40				Forte 40			280	248	528
Aprovação da Especificação dos Requisitos do Software pelo cliente	Médio 40	Médio 40		Fraco 20				Forte 40			140	220	360
Identificação das classes		Médio 40							Médio 8	Forte 20	68	68	136
Organização das classes		Médio 40							Médio 8	Forte 20	68	68	136
Identificação dos relacionamentos		Médio 40							Médio 8	Médio 8	56	56	112
Identificação dos atributos e herança		Médio 40							Médio 8	Forte 20	68	68	136

Realização dos casos de uso (análise)		Médio 40							Médio 8	Médio 8	56	56	112
Revisão da análise		Forte 100	Médio 40					Forte 40	Médio 8	Médio 8	196	156	352
Desenho arquitetônico		Médio 40							Forte 20	Forte 20	80	128	228
Desenho dos dados persistentes		Médio 40							Forte 20	Forte 20	80	128	228
Detalhamento das classes de desenho		Médio 40							Médio 8	Médio 8	56	104	160
Desenho das interfaces de usuário		Médio 40							Médio 8	Médio 8	56	136	192
Realização dos casos de uso (desenho)		Médio 40							Médio 8	Médio 8	56	88	144
Desenho das liberações		Médio 40							Médio 8	Médio 8	56	104	160
Revisão do desenho		Forte 100	Médio 40					Forte 40	Médio 8	Médio 8	196	204	400
Planejamento dos testes		Forte 100	Forte 100					Forte 40	Fraco 4	Fraco 4	248	244	492
Desenho dos testes		Forte 100	Forte 100					Forte 40	Fraco 4	Fraco 4	248	268	516
Implementação dos testes		Forte 100	Forte 100					Forte 40	Fraco 4	Fraco 4	248	268	516
Execução dos testes		Forte 100	Médio 40					Forte 40	Fraco 4	Fraco 4	188	268	456
Verificação do término dos testes		Médio 40	Médio 40					Médio 16	Fraco 4	Fraco 4	104	116	220
Balanço final dos testes		Fraco 20	Médio 40					Médio 16	Fraco 4	Fraco 4	84	72	156
Desenho detalhado		Médio 40							Médio 8	Médio 8	56	144	200
Revisão do desenho detalhado		Forte 100	Médio 40					Médio 16	Médio 8	Médio 8	172	204	376
Codificação		Médio 40							Médio 8	Médio 8	56	144	200
Revisão do código		Forte 100	Médio 40					Médio 16	Fraco 4	Fraco 4	164	196	360

Compilação		Médio 40							Fraco 4	Fraco 4	48	128	176
Testes de unidade		Forte 100	Médio 40					Médio 16	Fraco 4	Fraco 4	164	180	344
Integração		Médio 40							Fraco 4	Fraco 4	48	112	160
Análise do usuário								Médio 16			16	140	156
Análise das tarefas								Médio 16			16	140	156
Definição dos requisitos de usabilidade					Médio 16			Médio 16			32	140	172
Planejamento da usabilidade					Médio 16			Médio 16			32	40	72
Planejamento das avaliações de usabilidade								Médio 16			16	40	56
Execução das avaliações de usabilidade								Médio 16			16	40	56
Comunicação dos resultados das avaliações								Médio 16			16	40	56
Levantamento junto ao cliente (equipe marketing)					Médio 16			Médio 16			32	140	172
Levantamento junto aos usuários finais					Médio 16			Médio 16			32	140	172
Definição dos requisitos específicos do design gráfico					Médio 16			Médio 16			32	140	172
Criação do design								Médio 16			16	140	156
Alterações do design gráfico a pedido do cliente								Médio 16			16	116	132
Aprovação do design gráfico pelo cliente								Médio 16			16	116	132

Recorte das imagens											0	16	16
Montagem do esqueleto das interfaces de usuário											0	16	16
Levantamento das metas gerenciais	Médio 40			Médio 40							80	40	120
Estimativas da fase de elaboração	Forte 100			Médio 40	Forte 40						180	100	280
Elaboração da proposta de especificação	Médio 40										40	40	80
Planejamento do desenho	Forte 100	Fraco 20		Médio 40		Forte 40		Médio 16			216	120	336
Aprovação do Plano de desenvolvimento pelo cliente		Fraco 20						Fraco 8			28	60	88
Planejamento da qualidade	Médio 40	Fraco 20		Médio 40		Forte 40		Médio 16			156	60	216
Planejamento detalhado das liberações	Forte 100	Fraco 20				Forte 40		Médio 16			176	120	296
Revisão e modificação dos planos de desenvolvimento e da qualidade	Forte 100	Fraco 20		Fraco 20				Médio 16			156	120	276
Planejamento detalhado da transição	Forte 100	Fraco 20		Médio 40		Forte 40		Médio 16			216	120	336
Coleta de métricas sobre produtividade da equipe	Médio 40	Fraco 20			Forte 40	Médio 16	Forte 40				156	0	156
Armazenagem e análise dos dados colhidos	Forte 100	Fraco 20			Forte 40	Médio 16	Forte 40				216	0	216

Tabela 28 – Atividades x Objetivos do Processo

Bibliografia

[Ambler98] Scott W. Ambler. *Process Patterns: Building Large-Scale Systems Using Object Technology*. Cambridge University Press. Cambridge – UK, 1998.

[Ambler99a] Scott W. Ambler. *More Process Patterns: Delivering Large-Scale Systems Using Object Technology*. Cambridge University Press. Cambridge – UK, 1999.

[Ambler99b] Scott W. Ambler. *Completing the Unified Process With Process Patterns*. AmbySoft Inc., 1999.

[Ambler99c] Scott W. Ambler. *The Design of a Robust Persistence Layer for Relational Databases*. AmbySoft Inc., 1999.

[Ambler00] Scott W. Ambler. *Enhancing the Unified Process: Software Process for the Post-2000 (P2K) World*. Ronin International (White Paper). Março de 2000.

[Blaha+98] Michael Blaha e William Premerlani. *Object-Oriented Modeling and Design for Database Applications*. Prentice Hall, 1998.

[Blaha+98a] Michael Blaha e William Premerlani. *Implementing UML Models with Relational Databases*. Rational Software Corporation, 1998.

[Blaha+99] Michael Blaha e William Premerlani. *Object-Oriented Design for Database Applications*. Miller Freeman, 1999.

[Booch96] Grady Booch. *Object Solutions: Managing the Object-Oriented Project.* Addison-Wesley, Reading – MA, 1996.

[Booch98] Grady Booch. *The Visual Modeling of Software Architecture for the Enterprise*. Rational Software Coorporation, 1998.

[Booch+99] Grady Booch, Ivar Jacobson e James Rumbaugh. *The Unified Modeling Language User Guide.* Addison-Wesley. Reading – MA, 1999.

[Brooks95] Frederick P. Brooks, Jr. *The Mythical Man-Month*. Addison-Wesley. Reading – MA, 1995.

[Brown+97] Kyle Brown e Bruce Whitenack. *Crossing Chasms: a Pattern Language for Object-RDBMS Integration*. Technical Journal of Knowledge Systems Corporation, 1997.

[Carvalho01] Júnia G. Carvalho. *Praxis Mentor – Uma ferramenta de apoio à utilização de um processo de desenvolvimento de software*. DCC/UFMG (Dissertação de Mestrado). Belo Horizonte – MG, 2001.

[Conallen99a] Jim Conallen. *Modeling Web Applications Architectures with UML*. Communications of ACM, Vol. 42, No. 10. Outubro de 1999.

[Conallen99b] Jim Conallen. *Building Web Aplications with UML*. Addison-Wesley. Reading – MA, 1999.

[Constantine+99] Larry L. Constantine e Lucy A.D. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design.* Addison-Wesley. Reading – MA, 1999.

[Deboni97] José Eduardo Zindel Deboni. *Traduzindo objetos em bancos de dados relacionais*. Voxxel Consultoria de Sistemas, 1997.

[Fraternali99] Piero Fraternali. *Tools and Approaches for Developing Data-Intensive Web Applications: A Survey*. ACM Computing Surveys, Vol. 31, No. 3. Setembro de 1999.

[Graham+97] I. Graham, B. Henderson-Sellers and H. Younessi. *The OPEN Process Specification*. Addison-Wesley. London – UK, 1997.

[Henderson98] B. Henderson-Sellers. *The OPEN-Mentor Methodology*. Object Magazine. Setembro de 1998.

[Henderson+99] B. Henderson-Sellers, R. Dué, I. Graham. *Existing OO Process-Focussed Methods: A Critique of RUP and OPEN from a PM perspective*. TOOLS Workshop on "Project Management of Object-Oriented Developed Systems". Julho de 1999.

[Humphrey90] Watts S. Humphrey. *Managing the Software Process*. Addison-Wesley. Reading – MA, 1990.

[Humphrey95] Watts S. Humphrey. *A Discipline for Software Engineering*. Addison-Wesley. Reading – MA, 1995.

[Humphrey97] Watts S. Humphrey. *Introduction to the Personal Software Process*. Addison-Wesley. Reading – MA, 1997.

[Humphrey99] Watts S. Humphrey. *Introduction to the Team Software Process*. Addison-Wesley. Reading – MA, 1999.

[IEEE94] IEEE. *IEEE Standards Collection – Software Engineering.* IEEE. New York – NY, 1994.

[Isakowitz+95] Tomás Isakowitz, Edward A. Stohr, P. Balasubramanian. *RMM: A Methodology for Structured Hypermedia Design.* Communications of the ACM, Vol.38, No. 8. EUA. Agosto de 1995.

[Jacobson+99] Ivar Jacobson, James Rumbaugh e Grady Booch. *The Unified Software Development Process.* Addison-Wesley. Reading – MA, 1999.

[Burdman99] Jessica R. Burdman. *Collaborative Web Development: Stretegies and Best Practices for Web Teams*. Addison-Wesley. Reading – MA, 1999.

[Nielsen00] Jakob Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing. USA, 2000.

[McConnell96] Steve McConnell. *Rapid Development.* Microsoft Press, 1996.

[Paulk+95] M. C. Paulk, C. V. WEBER et alli. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley. Reading – MA, 1995.

[Paula+98a] Wilson de Pádua Paula Filho e Cláudio Ricardo Guimarães Sant'Ana. *Manual de Engenharia de Produtos de Software - Parte I: Recomendações*. DCC/UFMG (Relatório Técnico). Belo Horizonte – MG, 1998.

[Paula+98b] Wilson de Pádua Paula Filho e Cláudio Ricardo Guimarães Sant'Ana. *Manual de Engenharia de Produtos de Software - Parte II: Padrões e Modelos*. DCC/UFMG (Relatório Técnico). Belo Horizonte – MG, 1998.

[Paula+98c] Wilson de Pádua Paula Filho e Cláudio Ricardo Guimarães Sant'Ana. *Manual de Engenharia de Processos de Software - Parte I: Políticas*. DCC/UFMG (Relatório Técnico). Belo Horizonte – MG, 1998.

[Paula+98d] Wilson de Pádua Paula Filho e Cláudio Ricardo Guimarães Sant'Ana. *Manual de Engenharia de Processos de Software – Parte II: Padrões e Modelos*. DCC/UFMG (Relatório Técnico). Belo Horizonte – MG, 1998.

[Paula00] Wilson de Pádua Paula Filho. *Multimídia: Conceitos e Aplicações*. LTC Editora. Rio de Janeiro – RJ, 2000.

[Paula01] Wilson de Pádua Paula Filho. *Engenharia de Software: Fundamentos, Métodos e Padrões*. LTC Editora. Rio de Janeiro – RJ, 2001.

[Quatrani98] Terry Quatrani. *Visual Modeling with Rational Rose and UML*. Addison-Wesley. Reading – MA, 1998.

[Rational99] Rational Software Corporation. *Integrating Object and Relational Technologies*. Rational Software Corporation – Technical Support, 1999.

[Ribeiro99] Aloísio Antônio Ribeiro Júnior. *Um Estudo para a Introdução de Práticas de Engenharia de Usabilidade em uma Metodologia de Desenvolvimento de Software*. DCC/UFMG (Dissertação de Mestrado). Belo Horizonte – MG, 1999.

[Rossi96] Gustavo Hector Rossi. *Um Método Orientado a Objetos para o Projeto de Aplicações Hipermídia*. DI/PUC-RIO (Tese de Doutorado). Rio de Janeiro – RJ, 1996.

[Rumbaugh+99] James Rumbaugh, Ivar Jacobson e Grady Booch. *Unified Modeling Language Reference Manual.* Addison-Wesley, Reading – MA, 1999.

[Schwabe+98] Daniel Schwabe, Gustavo Rossi. *Developing Hypermedia Applications using OOHDM*. 1998.

[Schwabe+99] Daniel Schwabe, Patrícia Vilain. *Notação da Metodologia OOHDM*. PUC-RIO. Rio de Janeiro – RJ, abril de 1999.

[Vieira01] Fabiana Ruas Vieira. *e-Personal: Um Ambiente para Avaliar Estratégias de Personalização para Sítios WEB*. DCC/UFMG (Dissertação de Mestrado). Belo Horizonte – MG, 2001.

[Ward+99] Stan Ward, Per Kroll. *Building Web Solutions with Rational Unified Process: Unifying the Creative Design Process and the Software Engineering Process*. A Rational Software & Context Integration white paper – 1999.