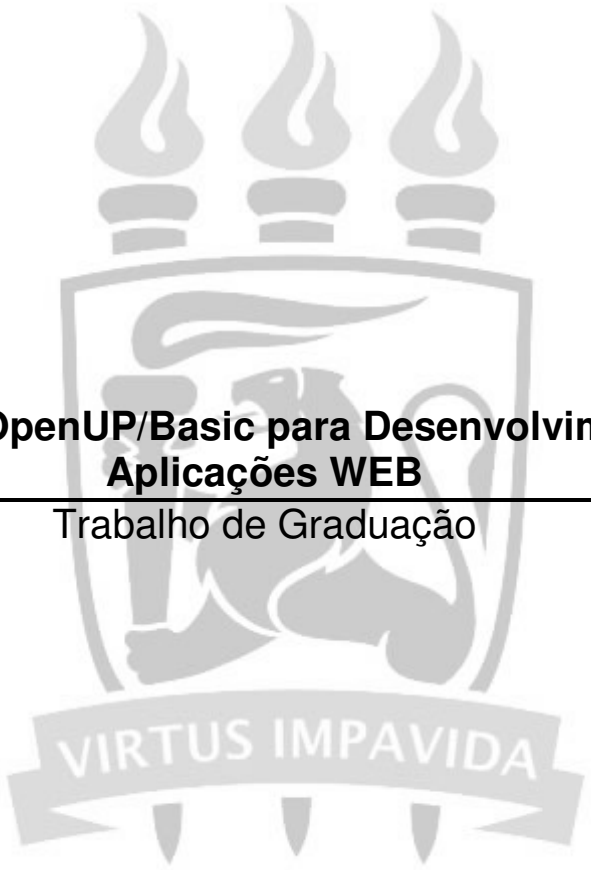
UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA



Utilizando OpenUP/Basic para Desenvolvimento de Aplicações WEB

Trabalho de Graduação

Aluno: Carlos Eduardo Albuquerque da Cunha (ceac@cin.ufpe.br)
Orientador: Alexandre Marcos Lins de Vasconcelos (amlv@cin.ufpe.br)

Recife, 29 de março de 2007.

Assinaturas

Este Trabalho de Graduação é resultado dos esforços do aluno Carlos Eduardo Albuquerque da Cunha, sob a orientação do professor Alexandre Marcos Lins de Vasconcelos, conduzido no Centro de Informática da Universidade Federal de Pernambuco. Todos abaixo estão de acordo com o conteúdo deste documento e os resultados deste Trabalho de Graduação.

_____

Alexandre Marcos Lins de Vasconcelos
Orientador

_____

Carlos Eduardo Albuquerque da Cunha
Graduando

"Our deepest fear is not that we are inadequate.
Our deepest fear is that we are powerful beyond measure.
It is our light, not our darkness
That most frightens us."

(Marienne Thompson)

Dedicado à meus pais Lourenço e Sonia,
pelo exemplo de dedicação e amor.
Dedicado à meus irmãos Débora e Pedro,
por todo o carinho e amizade durante os anos.
Dedicado à Bárbara,
pelo amor, compreensão e companheirismo.

Agradecimentos

Primeiramente tenho que agradecer a meus pais, Lourenço e Sonia. Vocês me deram todo o carinho e apoio que um filho poderia pedir, serei eternamente grato nunca esquecendo os exemplos de caráter que vocês são.

Agradeço a minha irmã Débora e meu irmão Pedro por todos os momentos maravilhosos que passamos juntos, toda a cumplicidade nas traquinagens e demonstração de amizade e apoio nas horas mais difíceis.

Agradeço a Bárbara por ser uma pessoa tão especial em minha vida, capaz de agüentar todo o meu mau humor e de quebra dedicar muita paciência, carinho, amor, compreensão e acima de tudo amizade.

Ao professor Alexandre Vasconcelos, meus sinceros agradecimentos, por ter aceitado participar da incrível empreitada de prestar orientação a este trabalho.

Aos mestres Alexandre Mota, Augusto Sampaio, Adnan "O Pai" Sherif, Ruy Guerra e Anjolina Grisi pelo exemplo de profissionais que foram meu muito obrigado.

Aos maloqueiros que me acompanham desde os tempos de Santa Maria: Gabriel, Breno, Márcio e Rodrigo com os quais já vão mais de dez anos de grandes amizades!!!

A Pedu, Marcu, Diego, Lulagi, Felipe, Brenna, Rafa, Thamine, Doloreta, Ceci, Rachel, Renatinha obrigado por tantas farras e pelas amizades desde as remotas aulas do terceiro ano.

A Niltu por todos os rodízios de sushi que enfrentamos, trocando idéias na busca da melhor forma de ganhar dinheiro. Ainda não ganhamos o dinheiro planejado mas pelo menos estamos no caminho certo!!!!

A Paulito, meu ídolo, por todas as vezes que fomos para o amarelinho tomar caldinho e filosofar, com certeza esta é uma das amizades que levarei para o resto da vida!

Aos meliantes Allan “Saci”, Thiago “O enroladinho”, Felipe “Acerola”, Fernando “Forró” Brayner, Gustavo “O veio”, Farley “O padim”, Zé Carlos “O ignorante”, Rafael “Cabelinho”, Rodrigo, Adelmário, Adriano, Martinelli, Livar, Arthur, Thierry, Bengt “Grande Ogro Escandinavo”, Sylvinha, Jeanne, Vitão, Tadeu “MTA”, Bruno “BPE”, Carlinhos “barba-ruiva”, Aninha, Carol, Williams, René, Carina, Pablo, João, Marcellus e Augusto por enfrentarem a batalha da graduação comigo e ainda por cima saindo vivos após tantas noites em claro trabalhando, ajudando uns aos outros

Agradeço a Joabe “O filho do senhor” Junior por ter passado por muitos apertos nos projetos da graduação e pela grande amizade que ganhei.

Agraceço a Paulo “Little Chicken” por esta amizade que nem sabemos como começou, mas que sempre rendeu bons momentos e muitas happy hours.

A galera da Icorp, principalmente Leandro, Bárbara e Caiozinho, que tiveram que agüentar todo meu mau humor e segurar as pontas dos projetos devido as minhas ausências.

Agradeço a Leonardo “menino de Surubim” Monteiro, Geraldo “O Pantera” Fernandes e minha brother Laís Xavier pelas aventuras de uma graduação muito louca e cheia de alegrias, além de terem encarado a comissão de formatura comigo, tendo um monte de trombadinhas enchendo o saco.

E finalmente a você leitor por ter chegado até esta parte da monografia só para saber de quem eu tirei sarro. Como diria o filósofo: “Aquele abraço” e até a próxima.

Resumo

Com o passar dos anos, e a disseminação da internet (World-Wide-Web), empresas de todas as partes do mundo levaram para a WEB uma gama de transações e processos para que clientes e funcionários estejam munidos de informações necessárias para a realização de operações vitais de seu cotidiano. Diferentemente dos softwares tradicionais, não existem processos que estudem os aspectos particulares do ciclo de desenvolvimento de aplicativos para WEB, portanto, este trabalho apresentará uma extensão do OpenUP/Basic que atenda a todas as necessidades do desenvolvimento de software para WEB.

Índice

Índice de Figuras

Índice de tabelas

Capítulo 1

Introdução

A Internet, durante o decorrer de sua história, tornou-se uma força na disseminação de informações entre universidades, órgãos governamentais e, posteriormente, a sociedade civil em todas as suas camadas. Seja na utilização de *e-mails* e *chats* ou em transações realizadas através de *e-commerce* ou e-*banking*, a Web mostra-se indispensável para realização das tarefas do dia-a-dia.

Com o aumento significativo da complexidade dos sistemas envolvidos e a importância das transações associadas a esses sistemas, os aplicativos para Web tendem a exigir maior cuidado em todas as fases de seu processo de desenvolvimento.

Diante deste cenário, cada vez mais complexo, este trabalho propõe o estudo e extensão, para o desenvolvimento de aplicações Web, das disciplinas de Requisitos e Análise e Projeto do processo de desenvolvimento de software OpenUP/Basic, o qual foi criado pela IBM e recentemente doado a Eclipse Foundation.

Este capítulo apresentará a motivação, o escopo e os objetivos deste trabalho, assim como sua estruturação.

1.1 Motivação

De acordo com [31] o crescimento do montante em operações de *e-commerce* na América Latina atingirá uma taxa de 40% ao ano, com manutenção desta até o ano de 2010. Números mais expressivos são apresentados no Brasil [32,33] onde as taxas de crescimento apresentadas são maiores e a recorrência de pedidos por clientes, os quais efetuaram mais de três compras nos últimos seis meses, atinge 33%.

Pensando nesses números, empresas de todos os ramos tendem a levar seus serviços e, principalmente, seus produtos à grande rede de computadores. Sendo os *e-commerces* os maiores exemplos do poderio financeiro da era virtual, outros tipos de serviços também vêm migrando gradativamente para o mundo dos computadores.

E-banking, *e-learning*, *e-government*, etc. são exemplos de aplicações que vêm ganhando espaço e adeptos dos mais variados segmentos da sociedade, principalmente no segmento empresarial, onde são vistos como diferenciais competitivos. Com isso a complexidade dos aplicativos voltados para a Internet cresce diretamente proporcional ao seu tamanho.

Devido ao alcance pretendido com as aplicações Web, as mesmas devem possibilitar uma imensa gama de funcionalidades a um grupo heterogêneo de usuários. Partindo deste princípio, o estudo das regras de negócio e execução das transações, incluindo a apresentação correta de informações, que obedeçam a essas regras, torna as aplicações Web bastante peculiares e cheias de possíveis desentendimentos entre *stackholders* e desenvolvedores.

Além das novas características apontadas, o aumento da importância deste novo tipo de aplicativo, faz com que os mesmos sejam alvo de constantes ataques e passíveis de possíveis fraudes. Neste último caso, podendo acarretar no fechamento de empresas e processos judiciais.

Como apresentado, os aplicativos Web possuem uma série de peculiaridades, em relação aos aplicativos convencionais, e, portanto, não possuem um suporte completo

dos processos de desenvolvimento de software presentes no mercado. Diante deste cenário faz-se necessário adaptar os processos de desenvolvimento a esta nova realidade.

1.2 Escopo e objetivos

Diante das novas características presentes no ambiente Web (sistemas em hipertexto e hipermídia, arquitetura distribuída, crescimento exponencial de usuários, etc. [17]), este trabalho pretende apresentar a extensão de um processo de desenvolvimento existente no mercado, para que o mesmo seja capaz de suprir as necessidades existentes em aplicativos Web. O processo escolhido foi o OpenUP/Basic [28], criado pela IBM [29] e recentemente incorporado a Eclipse Foundation [27].

Apesar de não possuir a amplitude, nem a disseminação no mercado, comparando-se com outros processos, como Rational Unified Process (RUP) [25] e Extreme Programming (XP) [24], não seria possível estender todo o OpenUP/Basic, então, foram escolhidos as disciplinas de Requisitos e Análise e Projeto para a atividade deste trabalho. O procedimento para a realização deste trabalho é relatado logo abaixo.

Inicialmente foi realizado um levantamento das características essenciais dos diversos tipos de aplicações Web. Este estudo visava mapear as necessidades dos diversos grupos de usuários desses sistemas. Tais necessidades podem ser definidas como o conjunto de requisitos funcionais e não-funcionais, apontados pelos usuários, que devem ser incorporados a quaisquer sistemas Web.

Em seguida, um estudo aprofundado do OpenUP/Basic foi realizado com o intuito de buscar todos os componentes importantes relacionado a este processo. Estes componentes, os quais serão definidos com o decorrer do trabalho, foram: princípios, disciplinas, fases, atividades e atores.

Com estas atividades realizadas, o passo seguinte foi verificar as possíveis alterações (extensões) no processo escolhido, deixando-o mais adequado a enfrentar os problemas relacionados com o desenvolvimento de aplicativos Web e por fim, permitindo que estes estejam, ao final do processo de desenvolvimento, de acordo com as exigências comuns aos respectivos usuários.

1.3 Estrutura do Trabalho

Além deste capítulo introdutório, este trabalho está dividido em mais 4 capítulos divididos da seguinte forma:

O Capítulo 2 apresenta uma introdução à Engenharia de Software para Web, demonstrando particularidades das aplicações Web, bem como as tecnologias utilizadas no desenvolvimento das mesmas.

No Capítulo 3, é introduzido o processo de desenvolvimento de software OpenUP/Basic, demonstrando seus atores, fluxos, subfluxos e atividades. O foco principal deste capítulo são os fluxos de requisitos e análise e projeto devido aos objetivos deste trabalho.

O Capítulo 4 apresenta todas as propostas de extensão para o OpenUP/Basic, para que o mesmo esteja melhor contextualizado ao desenvolvimento de aplicações Web.

Finalmente, o Capítulo 5, apresenta as conclusões deste trabalho ratificando as contribuições do mesmo e sugerindo caminhos para realização de trabalhos futuros.

Capítulo 2

Engenharia de Software para Web

Atualmente a Web é considerada um ambiente vital para disseminação e recuperação de informações. Sejam essas informações frutos de trabalhos de pesquisa ou relativas à movimentação financeira de empresas ou indivíduos, todos os usuários da grande rede aspiram acessar suas informações, ou de terceiros, indiferentemente de onde estejam ou da hora que irão acessar.

A Engenharia de Software para Web pretende estabelecer normas para suporte, gerenciamento e desenvolvimento de processos para desenvolvimento de softwares capazes de lidar com as particularidades dos aplicativos Web, e, acima de tudo, exigências de seus usuários.

Neste capítulo serão apresentados os enfoques da engenharia de software, um comparativo sobre aplicação Web versus aplicações tradicionais e um resumo das tecnologias utilizadas na construção de aplicativos Web.

2.1 Introdução à WEB

Em sua criação, durante a década de 1960, a internet (inicialmente chamada de APARTNET [8]) tinha como objetivo compartilhar informações entre unidades de pesquisa do exército americano. Com o sucesso inicial, vários serviços foram criados e incorporados nas décadas seguintes, tais como o e-mail, o acesso remoto e o compartilhamento de arquivos. Sendo na década de 1990, o surgimento de um dos mais difundidos o *World Wide Web* [7,10], simplesmente conhecido como Web.

A Web é baseada no paradigma de hipertexto [13], no qual usuários utilizam programas chamados *browsers* para acessar conteúdos disponibilizados em servidores ao redor do mundo. Apesar da força e solidez apresentada, recentemente o conceito de hipermídia [11,12,13] foi associado à Web, por causa da manipulação de conteúdos de multimídia.

Deste modo, a Web torna-se um poderoso instrumento na realização das atividades mais comuns do cotidiano, tendo elas os mais diferentes fins, de entretenimento a movimentações financeiras. Isso, em parte, resultado do alcance da internet, cujo acesso pode ser realizado de todos os pontos do planeta e, inclusive, com alcance espacial.

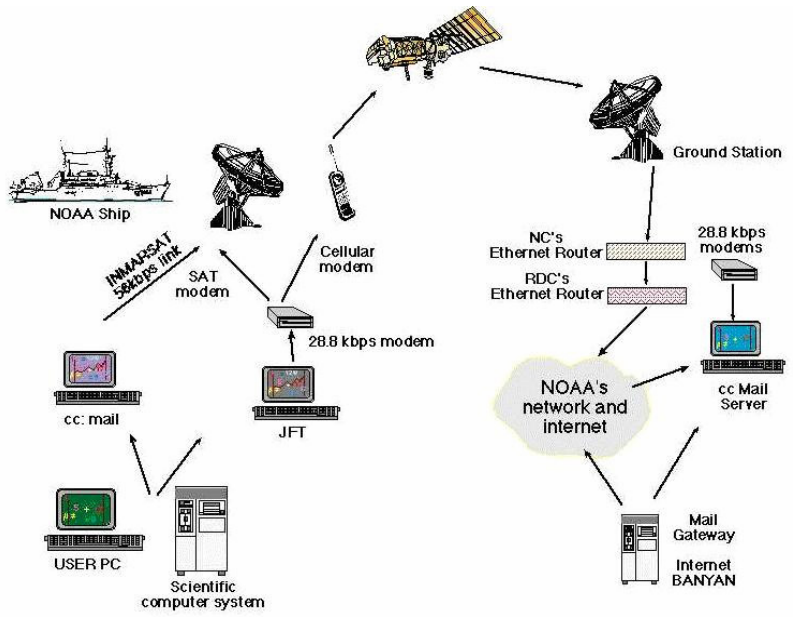


Figura 2.1 – Acesso a rede mundial de computadores.

Em razão dessas características apontadas, a Web torna-se o ambiente perfeito para que as empresas ampliem a oferta de serviços para os atuais clientes e consigam atingir um crescente número de potenciais novos clientes, que acharam neste ambiente desburocratizado o local perfeito para aquisição de informações vitais para suas decisões.

A tabela abaixo demonstra alguns dados sobre o crescimento do acesso a Web:

População mundial e utilização mundial da Web						
Região	**População (2007 Est.)**	**% Pop. mundial**	**Utilização da Web**	**% incluídos na Web**	**% Utilização mundial**	**Crescimento (2000-2007)**
África	933,448,292	14.2 %	33,334,800	3.6 %	3.0 %	638.4 %
Ásia	3,712,527,624	56.5 %	398,709,065	10.7 %	35.8 %	248.8 %
Europa	809,624,686	12.3 %	314,792,225	38.9 %	28.3%	199.5 %
Oriente Médio	193,452,727	2.9 %	19,424,700	10.0 %	1.7 %	491.4 %
América do Norte	334,538,018	5.1 %	233,188,086	69.7 %	20.9%	115.7 %
América Latina	556,606,627	8.5 %	96,386,009	17.3 %	8.7 %	433.4 %
Oceania	34,468,443	0.5 %	18,439,541	53.5 %	1.7 %	142.0 %
Total	**6,574,666,417**	**100.0 %**	**1,114,274,426**	**16.9 %**	**100.0 %**	**208.7 %**

Tabela 1 – População Mundial X Acesso a Web [14]

Tendo em vista o estudo apresentado na Tabela 1 e de todas as características apresentadas sobre a Web, podemos concluir que a Web demandará das empresas maciços investimentos em infra-estrutura e adaptação de seus serviços e processos a esta nova realidade, resultando em sistemas cada vez mais complexos.

2.2 Aplicativos Web

Essencialmente, aplicativos Web têm como objetivo realizar transações de negócio requisitadas pelos usuários, otimizando a experiência dos mesmos na conclusão de suas atividades.

Este tipo de aplicativo é alvo de grandes investimentos por partes de CIO's (*Chief Information Office*) na busca de novas oportunidades de negócios [30]. Dentre as

razões relacionadas para adoção desta tecnologia, encontramos três fatores fundamentais: alta popularidade da tecnologia entre clientes em potencial, baixo custo de investimento, e alta possibilidade de alcance dos objetivos pretendidos [17].

Existem três maneiras de disponibilizar o acesso de usuários a uma aplicação Web:

- *Intranet* – a aplicação será disponibilizada apenas a usuários de uma rede privada de computadores, ou rede locais (LAN).
- *Extranet* – a aplicação será disponibilizada através da extensão de uma determinada Intranet permitindo o acesso de usuários externos aos domínios da mesma.
- *Internet* – a aplicação será disponibilizada para qualquer usuário através da Internet.

Além dos aspectos apresentados acima, sobre investimentos e opções para disponibilizar as aplicações Web, faz-se necessário uma adaptação nos processos das empresas para que estas sejam capazes de usufruir de todos os benefícios que essa nova tecnologia proporcionará [17]. Outro aspecto importante é a mudança na cultura das empresas, principalmente em seu nível gerencial, que devem almejar os aplicativos Web como ferramentas alinhadas aos negócios proporcionando uma maior rentabilidade dos mesmos somados a melhora dos processos internos.

2.3 Aplicativos Web versus Aplicativos Tradicionais

Do ponto de vista da Engenharia de Software (gerenciamento, análise, desenvolvimento, suporte, etc.) aplicativos Web e aplicativos tradicionais são essencialmente equivalentes, entretanto diferenças com relação à multiplicidade de perfis e contextos, acesso concorrente a informações e tratamentos das mesmas, entre outras, são marcantes demonstrando a preocupação dos aplicativos Web em relação à navegabilidade através do sistema.

Alguns estudos [15,21] classificam os diferentes aspectos desta perspectiva sobre navegabilidade em três diferentes grupos: Navegação, Organização da Interface e Implementação.

Referentes à Navegação.

- O que constitui uma "unidade de informação" com respeito à navegação?
- Como estabelecer *links* significativos entre unidades de informação?
- Onde o usuário começará sua navegação?
- Como organizar o "espaço navegacional", estabelecendo possíveis seqüências de unidades de informação a qual o usuário navegará?
- Caso uma interface Web esteja sendo adicionada a um sistema existente, como mapear objetos de dados existentes em unidades de informação, e quais relacionamentos serão transformados em *links*?

Referentes à Organização da Interface

- Quais os objetos da interface serão perceptíveis ao usuário? Como será o relacionamento entre esses objetos e os objetos de navegação?
- Qual será o comportamento da interface à medida que o usuário for utilizando-a?
- Como as operações de navegação serão distinguidas das operações de interface e das operações de processamento?
- Como o usuário será capaz de localizar-se no espaço navegacional?

Referentes à Implementação

- Como as unidades de informação são mapeadas sobre páginas?
- Como operações de navegação são implementadas?
- Como objetos de navegação são implementados?
- Como bases de dados são integradas a aplicação?

Este foco em navegabilidade faz com que aplicativos Web tornem-se cada vez mais presentes nas atividades dos usuários da Web, sendo este último em grande expansão.

2.4 Engenharia de Software para Web

Com a disseminação da Web e o crescimento da complexidade dos aplicativos Web, além do impacto dos mesmos nos vários setores da sociedade, existe a necessidade de formalizar toda a estrutura para desenvolvimento das aplicações Web, isto implica em estabelecer métricas para gerenciamento, desenvolvimento e suporte garantindo qualidade e integridade de todos os aspectos envolvidos em sua produção.

A complexidade dos aplicativos Web está relacionada a uma série de fatores que obrigatoriamente devem ser tratados para o bom funcionamento e manutenção desses sistemas. Em [16] um conjunto desses fatores é apresentado:

- Ambientes heterogêneos de hardware e software;
- Aspectos relacionados a requisitos, principalmente não-funcionais como concorrência, distribuição, segurança, escalabilidade, persistência, etc.;
- Tipos ilimitados de perfis de usuários utilizando diferentes *browsers* e protocolos, além das mais variadas plataformas (computadores, PDAs, celulares, etc.);
- Arquitetura dos sistemas deve ser flexível e estruturada devido a constantes alterações nos sistemas e integrações com outros sistemas Web e sistemas legado;
- Conteúdos dos sistemas Web podem ser dinâmicos, com freqüentes atualizações;
- Sistemas Web terão de gerenciar diversos tipos de perfis de usuários com possíveis implementações de funcionalidades específicas.

Infelizmente, grande parte das aplicações Web existente utilizaram metodologias de desenvolvimento *ad-hoc* [34], o que impossibilita melhor acompanhamento na evolução desses aplicativos e inviabiliza estudos comparativos e obtenção de métricas para novos projetos. Esses projetos dependeram da qualidade e demandaram maior esforço das pessoas envolvidas para realização de todas as tarefas, isto comprova a necessidade de processos definidos para orientar o desenvolvimento [35].

Visto que a ausência de procedimentos formais aumenta consideravelmente a possibilidade de falhas ocorrerem, a procura por processos formalizados para acompanhamento de todas as etapas do desenvolvimento de aplicativos pretende minimizar a ocorrência dessas falhas nos sistemas, evitando diversos tipos de danos e prejuízos. A ocorrência de falhas críticas pode diminuir a confiança na Web causando a chamada "crise da Web" [34,35] que seria mais difundida do que a crise do software [36].

Para evitar uma possível "crise da Web", faz-se necessário a utilização de ferramentas, métricas, processos, metodologias, etc. para garantir que as aplicações Web consigam adaptar-se facilmente a este ambiente extremamente heterogêneo decorrente da utilização de múltiplas mídias e sistemas operacionais e a presença de um grande e complexo conjunto de perfis de usuários.

A Engenharia de Software para Web deve estabelecer metodologias que acompanhem todo o desenvolvimento de softwares, garantindo a qualidade dos produtos. Seja na elaboração de novos processos ou na adaptação de processos definidos pela Engenharia de Software tradicional, o objetivo é possibilitar as equipes de desenvolvimentos metodologias que abranjam todas as atividades do novo projeto, desde a concepção do sistema, incluindo também o desenvolvimento, avaliação de qualidade e desempenho, até a fase de manutenção.

Por esses motivos, a Engenharia de Software para Web possui uma série de aspectos que deverão ser assistidos e evoluídos. Abaixo segue um grupo de tópicos que deverão ser trabalhadas [35]:

- Especificação e análise de requisitos;
- Metodologias, técnicas e processos de desenvolvimento de sistemas para Web;
- Integração com sistemas legados;
- Migração de sistemas legados para o ambiente Web;
- Gerência de informações;
- Desenvolvimento de aplicações Web de tempo real;
- Técnicas e ferramentas de teste, verificação e validação;

- Avaliação, controle e garantia de qualidade;
- *Web metrics* – métricas para estimativas de esforço de desenvolvimento;
- Especificação e avaliação de performance;
- Manutenção, atualização e integração;
- Desenvolvimento centrado no usuário, envolvimento e *feedback* do usuário;
- Preocupações legais e de privacidade;

Com todos os aspectos apresentados acima, este trabalho estará focado nos processos e metodologias para desenvolvimento de softwares para Web. Isso será apresentado no capítulo 4 quando apresentaremos a extensão dos fluxos de requisitos e análise e projeto do OpenUP/Basic.

2.5 Projetos de sistemas Web

Diante das exigências relacionadas a navegabilidade das aplicações Web, uma série de metodologias foram criadas. A maior parte dessas metodologias concentra-se nos fluxos de análise e projeto na tentativa de solucionar problemas provenientes da complexidade navegacional requerida pelos aplicativos Web. Entre as mais conhecidas temos HDM [38,40], RMM [39], OOHDM [21,41] e RNA [37], visto que não é objetivo deste trabalho descrever as diferentes metodologias, apenas uma breve descrição será apresentada nas subseções a seguir.

2.5.1 Hypermedia Method Design (HDM)

Uma das primeiras metodologias utilizadas na definição de estruturas de aplicativos de hipermídia. Baseada nas notações de E-R (entidade-relacional), apresentando uma extensão para incorporar conceitos como nós e *links* na construção de modelos de navegação.

As entidades presentes nos modelos de HDM possuem estruturas internas com semânticas definidas que especificam a navegabilidade e a forma com que as informações são disponibilizadas aos usuários.

Segundo [38], modelagens de aplicativos Web baseadas em HDM devem expor alguns aspectos básicos, tais como:

- *Conteúdo* – todo tipo de informação envolvida na aplicação;
- *Estrutura* – organização da estrutura do conteúdo apresentado;
- *Apresentação* – define como as informações e funcionalidades são exibidas ao usuário;
- *Dinâmica* – defina a interação entre o aplicativo e o usuários (operações passíveis de serem realizadas);
- *Interações* – composição da dinâmica de operações e apresentação de conteúdo.

2.5.2 Relationship Management Methodology (RMM)

Nesta metodologia o design das aplicações é representado através do RMDM (Relationship Management Data Model) que é baseado nos modelos HDM e E-R (entidade-relacional). Utilizando esta metodologia, a construção das aplicações de hipermídia deverá seguir sete etapas:

- *Entity-relational design* – mapeamento das entidades, atributos e relacionamentos existentes na aplicação;
- *Slice-design* – nesta etapa acontece o agrupamento das entidades para apresentação, onde os *slices* serão considerados diferentes "unidades de apresentação";
- *Navigational design* – etapa de identificação dos caminhos de navegação, onde esta é realizada através de *links*, menus, *index*, *guided tour* e *indexed quided tour*;
- *User interface design* – durante esta etapa todos os *layouts* são definidos, incluindo design de *links*, menus, índices, etc.;
- *Protocol conversion design* – nesta etapa acontece a conversão entre componentes de RMM em objetos presentes nas aplicações de hipermídia;

- *Run-time behavior* – a definição do comportamento da aplicação perante a interação com o usuário é definida nesta etapa;
- *Construction and testing* – fase de desenvolvimento e teste da aplicação definida através do processo.

2.5.3 Object-Oriented Hypermedia Design Method (OOHDM)

Esta metodologia é baseada em um modelo iterativo, incremental e orientada a protótipos, utilizando modelos OO (orientados a objeto) os quais são construídos a cada iteração.

O processo de desenvolvimento através desta metodologia possui quatro atividades básicas:

- *Conceptual modeling* – nesta modelagem é apresentada a aplicação como um diagrama de classes, onde as classes aqui apresentadas são chamadas de classes navegacionais;
- *Navegational modeling* – esta atividade introduz o conceito de "contexto navegacional" onde a toda a estrutura de navegação da aplicação é apresentada. Através deste conceito são definidas as permissões de transição de objetos navegacionais através de diferentes contextos navegacionais;
- *Abstract interface design* – este modelo representará o conjunto de interfaces e objetos passíveis visualizados pelos usuários da aplicação. OOHDM utiliza *Abstract Data Views* (ADVs) [42] para mapear aspectos estáticos da interface e *Statecharts* [43] para mapear aspectos dinâmicos;
- *Implementation* – fase de desenvolvimento da aplicação. Foi desenvolvido o OOHDM-Web que auxilia na prototipação gerando páginas *templates*.

2.5.4 Relationship-Navegational Analysis (RNA)

Esta metodologia está focada para a análise de aplicações, principalmente sistemas legados, e define um procedimento próprio que deverá ser obedecido durante sua utilização no processo de desenvolvimento da nova aplicação.

O procedimento utilizado é descrito abaixo:

- *Stakeholder analysis* – identificação das pessoas, não apenas usuários, relacionados com o sistema;
- *Element analysis* – listagem dos elementos relacionados com o projeto (documentos, modelos, formulários, etc.);
- *Relationship and metaknowledge analysis* – identificação de esquemas, processos, operações e relacionamentos de qualquer tipo.
- *Navigation analysis* – identificação de estruturas de navegação e de acesso da aplicação;
- *Relationship and metaknowledge implementation analysis* - Escolha de quais relacionamentos mapeados anteriormente serão implementados.

2.6 Principais Tecnologias

Como apresentado, as aplicações Web são bastante dinâmicas, também exigindo que as tecnologias utilizadas em seu desenvolvimento sejam capazes de executar em diversos ambientes e, acima de tudo, possibilitar uma fácil manutenção.

Abaixo segue uma lista das tecnologias mais difundidas para desenvolvimento de aplicativos Web:

- *Java*: Linguagem de programação de âmbito geral, multi-thread, orientada a objeto, independente de plataforma [1]. Devido a essas características, tornou-se a linguagem de programação mais utilizada pelo mercado;

- *Servlets*: Programa Java utilizado nos servidores de aplicação que processa a requisição dos usuários, gerando conteúdo dinâmico e maior interação com o usuário. Baseado no paradigma de request-response [2];

- *Java Database Connectivy* (JDBC): Biblioteca da linguagem Java utilizada para acesso de recursos de banco de dados;

- *Enterprise Java Beans* (EJB): Componentes de arquitetura para sistemas desenvolvidos na tecnologia J2EE [1];

- *Struts*: Framework de código aberto para desenvolvimento de aplicações WEB, baseadas na estrutura Servlet/JSP [3];

- *Hibernate*: Framework de código aberto [4], implementado em Java, utilizado para acesso de base de dados. Utiliza mapeamento objeto relacional para melhor integração com aplicativos Java;

- *Java Server Pages* (JSP): Tecnologia desenvolvida para apresentação dos dados processados nas Servlets em página Web dinâmicas;

- *Tomcat*: Servidor para aplicações WEB desenvolvidas em Java. Software livre pertencente a Apache Foundation [5,6].

2.7 Considerações Finais

Neste capítulo foram apresentadas todas as características básicas envolvidas com o processo de desenvolvimento de software para Web. Introduzimos os alicerces da Engenharia de Software para Web, que pretende estabelecer normas para os processos de gerenciamento, desenvolvimento e suporte ao desenvolvimento de sistemas Web. Além disso, revisamos algumas das tecnologias utilizadas para desenvolvimento e suporte de aplicativos Web produzidos atualmente.

Em se tratando de processos de desenvolvimento, o próximo capítulo apresentará o OpenUP/Basic, cujas características serão posteriormente estendidas para adaptarem-se as exigências da Engenharia de Software para Web.

Capítulo 3

OpenUP/Basic

Com o aumento no número de novas aplicações Web e na adaptação de muitos outros sistemas legados para esta nova plataforma, faz-se necessário a criação ou adaptação de processos que estejam aptos a enfrentar problemas típicos do ambiente Web tais como gerenciamento de múltiplos usuários e projetos navegacionais complexos.

O OpenUP/Basic foi desenvolvido pela IBM com base nos processos RUP e XP na expectativa de reunir as melhores características de cada uma dessas metodologias. A estrutura deste processo torna-o perfeito para este trabalho visto que uma de suas propriedades é de ser passível a extensões, objetivo principal deste trabalho que é a adaptação de processos ao desenvolvimento de aplicativos Web.

Este capítulo apresentará todo o processo OpenUP/Basic incluindo seus princípios, áreas de conteúdo, atores e fases de iterações.

3.1 Visão Geral

A maioria dos projetos de desenvolvimento de software possui tempo médio de duração de três a seis meses com equipes variando entre três e dez pessoas. Podemos perceber que este tipo de projeto não comporta a robustez de processos de desenvolvimento como o RUP [25] e, em sua maioria, não utiliza todos os recursos presentes em processos de desenvolvimento como o XP [24].

Equipes de projetos de pequeno porte também devem manter o foco de suas atividades na entrega do produto requisitado pelos *stackholders* evitando sobrecarga de atividades e pondo em risco o andamento e futuro do projeto.

Com o propósito de solucionar os problemas apontados acima, a IBM [29] desenvolveu o OpenUP/Basic [28,44,45,46,47]. Este processo combina os aspectos positivos do desenvolvimento ágil [24] e do *Unified Process* [25] como ciclo de vida estruturado, desenvolvimento em iterações, gerenciamento de riscos, desenvolvimento baseado em casos de uso e cenário, entre outros. Esta combinação de características enaltece a aspectos dentre a equipe de desenvolvimento, como colaboração, garantindo benefícios aos *stackholders* minimizando o formalismo e improdutividade.

O OpenUP/Basic define um conjunto compacto de atores, tarefas e artefatos, onde atores realizam tarefas que utilizam e produzem artefatos. Este processo de desenvolvimento de software é iterativo podendo ser definido como:

- *Compacto* – Apenas conteúdos fundamentais para utilização do processo estão definidos;
- *Completo* – Significa que o processo abrange todas as fases de do ciclo de vida de desenvolvimento de software;
- *Extensível* – Este processo pode ser utilizado na forma como foi definido, entretanto permite que novos conteúdos de processos (fluxos, subfluxos, atores, etc.) sejam adicionados para melhor atender as características de um determinado projeto.

A Seção 3.2 detalha os princípios do processo em questão, em seguida, na seção 3.3, são apresentadas as áreas de conteúdo existentes. Com os princípios e áreas de conteúdo definidos, a seção 3.4 apresenta as disciplinas envolvidas no OpenUP/Basic para que em seguida sejam definidos os atores, atividades e as fases do ciclo de desenvolvimento nas seções 3.5, 3.6 e 3.7 respectivamente.

3.2 Princípios

O OpenUP/Basic está baseado em quatro princípios: Colaboração, Equilíbrio, Foco e Evolução. Esses princípios definem o processo por completo através da participação dos atores envolvidos, dos artefatos a serem produzidos e das tarefas a serem cumpridas no decorrer do ciclo de vida de desenvolvimento. Cada um desses princípios será detalhado nas subseções seguintes.

3.2.1 Colaboração

Softwares são desenvolvidos por pessoas com diferentes estilos e habilidades que trabalham em conjunto para que o produto final esteja de acordo com o especificado inicialmente. Portanto faz-se necessário a criação de atividades que mantenham o ambiente saudável possibilitando o entendimento do projeto e alinhamento das expectativas e interesses de todos os participantes (equipe de desenvolvimento, *stackholders*, consumidores, etc.). Dentre as principais atividades temos:

- *Entendimento comum* – Participantes do projeto devem manter um entendimento comum dos aspectos relacionados ao projeto, eles devem ser encorajados a exercitar uma comunicação pró-ativa com objetivo de conciliar seus interesses com de outros participantes e interesses do projeto;
- *Ambiente de alta confiabilidade* – Membros de equipes que não trabalham neste tipo de ambiente tendem a não expor suas idéias, interesses e passam a não ter iniciativa, causando possíveis atrasos de projeto. É de responsabilidade de todos os membros do projeto a manutenção de um ambiente confiável e agradável para a execução do trabalho.

- *Compartilhar responsabilidades* – Responsabilidades primárias devem ser atribuídas a cada membro da equipe, entretanto todos os membros possuem responsabilidade sobre a produção global no decorrer do projeto. Isto significa que todos devem estar dispostos a ajudar e serem ajudados na execução de atividades atrasadas com pessoas que passam por dificuldades em exercer suas funções.
- *Aprendizado contínuo* – Projetos de softwares são realizados por grupos de pessoas com habilidades técnicas e interpessoais variadas que devem estar em constante aprimoramento. É necessária a troca de experiência entre membros do projeto, pois todo aprendizado influenciará positivamente na resolução de situações adversas que venham a ocorrer a membros da equipe.

3.2.2 Equilíbrio

Membros de projeto e *stackholders* devem trabalhar para que o produto desenvolvido beneficie os *stackholders* da melhor maneira possível diante das adversidades previstas no projeto. Atingir equilíbrio no processo de desenvolvimento é um procedimento dinâmico visto que com o passar do tempo equipe e *stackholders* aprendem mais sobre o sistema que está sendo desenvolvido.

Para o sucesso ser atingido equipe e *stackholders* devem convergir seus entendimentos em três fatores: problemas tratados pelo sistema, adversidades da equipe de projeto (custo, cronograma, etc.) e adversidades da solução. Coletivamente esses fatores tornam-se os requisitos para o desenvolvimento do sistema.

O desafio das equipes de projeto é desenvolver soluções que maximizem as experiências dos usuários. O equilíbrio estará na escolha do melhor custo-benefício em *trade-offs* entre funcionalidades requisitadas e decisões de design na construção de arquitetura. Durante o processo de desenvolvimento o ponto de equilíbrio do projeto pode ser alterado devido a diversos fatores, entre eles temos novas oportunidades, novas funcionalidades desejadas, riscos contornados novos riscos.

Stackholders e desenvolvedores devem estar preparados para reavaliar compromissos e expectativas ajustando planos de acordo com a evolução do sistema. Abaixo segue algumas atividades para auxilio na busca do melhor equilíbrio:

- *Conhecer os stackholders* – Os melhores *trade-off* só serão realizado quando for de conhecimento quem são os *stackholders* do projeto e o realmente desejam. Se esta pré-condição for satisfeita, devem-se manter canais de comunicações abertos e em constante interação para que a equipe sempre saiba de suas reais necessidades;
- *Separar o problema da solução* – Pessoas são ensinadas a resolver problemas e não defini-los, entretanto, é crucial entender os problemas enfrentados para que os melhores *trade-offs* sejam efetuados. Detalhar os problemas facilitará na busca de caminhos para solucioná-los;
- *Utilizar cenários e casos de uso para obter requisitos* – Esta técnica permitirá obter e documentar requisitos funcionais de uma maneira de fácil entendimento para *stackholders* e usuários. Requisitos não-funcionais como usabilidade, desempenho, etc. poderão ser documentados utilizando técnicas tradicionais.
- *Estabelecer e manter prioridades* – Decisões ruins em estabelecer qual funcionalidade deverá ser implementada podem acarretar em atraso e cancelamento de projetos. Portanto a implementação de requisitos deve ser trabalhada em conjunto com *stackholders* durante todo o processo de desenvolvimento. Boas decisões agregam valor ao produto entregue e diminuem os riscos.
- *Gerenciar escopo* – Alterações de projetos são inevitáveis visto que as necessidades dos *stackholders* variam durante o ciclo de vida do projeto. Processos modernos sempre gerenciam alterações, constantemente adaptando-se a mudanças no ambiente e necessidades dos *stackholders*.

3.2.3 Foco

Sistemas com arquitetura mal definida evoluirão de forma ineficiente e autodestrutiva. Esse tipo de sistema dificulta evolução, reuso e integração com outros sistemas sem substanciais re-trabalhos. Além disso, a comunicação entre membros da

equipe de desenvolvimento torna-se ineficiente. O foco do desenvolvimento na arquitetura permitirá que desenvolvedores realizem melhores decisões técnicas garantindo uma evolução eficiente do sistema. Abaixo segue um conjunto de atividades sugeridas:

- *Criar arquitetura baseada nos conhecimentos atuais* – Crie uma arquitetura que representa apenas as necessidades atuais dos *stackholders*, dando flexibilidades e velocidade para os requisitos que são conhecidos atualmente. Especular requisitos futuros deixará sua arquitetura mais pesadas e propensa a re-trabalho;
- *Aumentar a abstração* – Sistemas podem ficar grandes e complexos dificultando a visualização do projeto por diversos membros da equipe, portanto utilize modelos de abstração focados nos problemas essenciais do projeto o que facilitará o entendimento do projeto por diversas perspectivas;
- *Problemas indicam a solução* – Deve ser mais difícil manter e adaptar as necessidades dos *stackholders* do que novas tecnologias. Portanto organize a arquitetura em função das necessidades do projeto.
- *Alta coesão e baixo acoplamento* – Alto acoplamento deixa o sistema frágil, com difícil entendimento e impossibilitando o reuso de componentes. Organize a arquitetura do sistema em componentes com alta coesão e baixo acoplamento, isto aumentará a compreensão, flexibilidade e oportunidades para reuso;
- *Reuso* – Desenvolvedores geralmente não utilizam reuso devido a componentes que não disponibilizam o que necessitam ou são de baixa qualidade. Entretanto, reuso garante ganho de tempo na realização de trabalhos, mesmo se os componentes já existentes tenham que ser adaptados ou distorções na arquitetura precisem ser realizadas.

3.2.4 Evolução

Na maioria das vezes é impossível obter todas as necessidades dos *stackholders*, por isso é necessário que todos na equipe estejam cientes dos riscos do projeto, das tecnologias utilizadas e possuam bom entendimento entre si. Mesmo que a equipe

saiba todas as necessidades, é bastante provável que elas mudem com o decorrer do ciclo de vida de desenvolvimento. Portanto é necessário dividir o projeto em pequenas iterações para demonstrar agregação de valor incremental entre iterações e obtendo constantes *feedbacks* por parte dos *stackholders*.

A intenção deste princípio é constantemente receber *feedbacks* por parte dos *stackholders* durante a evolução do ciclo de vida de desenvolvimento garantindo melhor gerenciamento dos riscos do projeto. Identificando e tratando os riscos garante melhor qualidade do processo e do produto entregue.

Não apenas o produto evolui, mas também a equipe encontra melhores maneiras de trabalhar em conjunto e buscar soluções. Abaixo segue uma lista de atividades relacionadas a este princípio:

- *Desenvolvimento em iterações* – Estratégia iterativa permite retorno constante das necessidades de projeto e possibilita disponibilizar funcionalidades aos *stackholders* para avaliação. Esta troca de idéias constante entre equipe e *stackholders* aumenta a qualidade do produto entregue gerenciando melhor os riscos de projeto e atendendo as expectativas do mesmo.
- *Gerenciamento de riscos* – Levantamento de riscos tardio pode gerar design de arquitetura ruim, escolha errada de tecnologias e implementação de requisitos não condizentes com as necessidades dos *stackholders*. Essa atividade deve ser realizada desde as primeiras fases do projeto e constantemente reavaliada para identificação de novos riscos e reclassificar prioridades.
- *Gerencie alterações* – Alterações de projeto são inevitáveis principalmente se vão de encontro às necessidades dos *stackholders*. Alterações tardias podem resultar em produtos de má qualidade e aumento no esforço necessário para realizá-las. Portanto sempre que alterações sejam requisitadas é necessário documentá-las e informar aos *stackholders* possíveis impactos destas alterações.
- *Avaliação objetiva de progresso* – Estabelecer um conjunto de métricas apontará as falhas e os sucessos durante o andamento do projeto. Isto

significa acompanhamento das atividades e impacto das alterações realizadas, gerenciando os riscos que por ventura possam diminuir a qualidade do produto entregue e não atendam as necessidades dos *stackholders*.

- *Reavaliação contínua do que é realizado* – Reuniões regulares da equipe de desenvolvimento identificarão riscos e problemas. Esta atividade pode ser realizada diariamente quando os membros dividem os status e responsabilidades individuais. Ao final de cada iteração é possível identificar áreas para melhoria para entrega ao final da próxima iteração.

3.3 Áreas de conteúdo

O Conteúdo do OpenUP/Basic é organizado em quatro grandes áreas de conteúdo também conhecidas como subprocessos, são elas: Colaboração e Comunicação, Propósito, Gerenciamento e Solução. Essas áreas de conteúdo estão entrelaçadas com as diferentes fases do desenvolvimento, suas atividades e tarefas, e são influenciadas pelos diferentes princípios anteriormente definidos.

Cada área de conteúdo define aspectos importantes da participação dos atores no processo de desenvolvimento. Nas subseções seguintes detalharemos cada área de conteúdo.

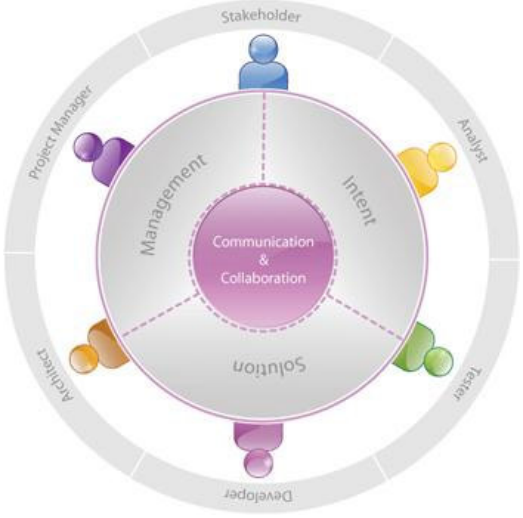


Figura 3.1 – Áreas de Conteúdo

3.3.1 Colaboração e Comunicação

Baseada no princípio de *Colaboração*, esta área de conteúdo reflete a natureza colaborativa do processo, envolvendo todos os tipos de atores presentes (serão definidos mais adiante na seção 3.5) com o objetivo primário de reunir as necessidades dos *stackholders*, desenvolver a solução e gerenciar o projeto.

Esta área de conteúdo expressa a habilidade de auto-organização das equipes, permitindo que diferentes atores compreendam as perspectivas relacionadas a seus cargos e quais as suas responsabilidades dentro do processo de desenvolvimento.

3.3.2 Propósito

Este subprocesso é baseado no princípio do *Equilíbrio* e representa como os propósitos apontados pelos *stackholders* são transmitidos aos desenvolvedores, assegurando que *builds* desenvolvidos sejam entregues, atendendo as expectativas e necessidades desses mesmos *stackholders*. Isto é feito através da comunicação da visão dos *stackholders* através de casos de testes e requisitos repassados para todos os membros da equipe.

A área de conteúdo de Propósito é construída baseada no subprocesso de Colaboração, definido na subseção anterior. Os impactos da camada de Colaboração neste subprocesso são:

- Todos os atores definidos no OpenUP/Basic que estão envolvidos no desenvolvimento da solução devem entender todas as necessidades dos *stackholders*;
- Todas as melhores práticas de desenvolvimento colaborativo devem ser utilizadas como guia para atividades relacionadas com esta camada.

A área de conteúdo de Propósitos foi definida para que todas as organizações possam alterá-la para que melhor atendam seus estilos de colaboração com os *stackholders*, sem que necessariamente impactem as áreas de conteúdo de Gerenciamento e Solução.

3.3.3 Gerenciamento

Esta área de conteúdo tem como principal atividade o gerenciamento de projeto, que envolve atividades como planejamento de projetos, planejamento de iterações, acompanhamento dia-a-dia de desenvolvimento de iterações, entre outras, e possui como base o princípio da *Evolução*.

Deve ser salientado que esta área de conteúdo é baseada no subprocesso de Colaboração (definida na subseção 3.3.2) e devido a esta influencia, determina que gerentes devam envolver toda a equipe no planejamento do projeto garantindo que:

- O conhecimento de todos esteja refletido no planejamento;
- Todos os membros da equipe realizem estimativas das suas atividades;
- Cada membro de equipe deve voluntariamente escolher as tarefas que deseja realizar, ao invés serem alocados a qualquer tarefa.

Assim como a área de conteúdo de Propósitos, este subprocesso foi definido de forma que qualquer organização possa adaptá-lo ao seu estilo de desenvolvimento, sem que ocorram impactos nas demais áreas de desenvolvimento.

3.3.4 Solução

Esta área de conhecimento é baseada no princípio de *Foco* e descreve todos os aspectos relacionados com a criação da arquitetura, design, implementação e teste de uma aplicação. Este subprocesso representa um guia para realização das seguintes atividades:

- Definição da arquitetura;
- Determinar viabilidade da arquitetura proposta;
- Desenvolvimento da arquitetura para design, implementação e teste de grandes alterações de projeto;
- Design, implementação e teste de pequenas alterações de projeto;
- Implementar e testar alterações triviais de projeto;
- Teste e validação de *builds* com qualidade aprimorada.

Esta área de conteúdo é baseada no subprocesso de Colaboração (definido na subseção 3.3.2) e garante que:

- Todos os atores definidos no OpenUP/Basic estão envolvidos no desenvolvimento da solução;
- Validação de *builds* é de responsabilidade de todos os membros da equipe;
- Melhores práticas de colaboração auxiliam no desenvolvimento da solução.

O subprocesso de Solução foi desenvolvimento para que qualquer organização possa adaptá-lo ao respectivo estilo de desenvolvimento sem que sejam necessárias alterações nas áreas de conteúdo de Gerenciamento e Propósitos.

3.4 Disciplinas

Disciplinas agrupam diversas tarefas e representam os diferentes aspectos do desenvolvimento de aplicações. No OpenUP/Basic é apresentado um conjunto de seis disciplinas, são elas: Análise e Design, Gerenciamento de Configurações e Alterações, Implementação, Gerenciamento de Projeto, Requisitos e Testes. Todas as disciplinas serão detalhadas nas subseções a seguir.

3.4.1 Análise e Design

Esta disciplina agrega tarefas que auxiliarão na criação do design futuramente implementado por desenvolvedores a partir dos requisitos do sistema. Possui as seguintes tarefas: *Analyze Architectural Requirements*, *Demonstrate the Architecture*, *Design the Solution* e *Develop the Architecture*.

Tem como principais objetivos transformar requisitos em design do futuro sistema, desenvolver uma arquitetura robusta para a solução e adaptar o design as características do ambiente de desenvolvimento.

A disciplina de Análise e Design está relacionada com outras disciplinas da seguinte maneira:

- *Requisitos* – produzem artefatos primários para execução das tarefas de Análise e Design;
- *Implementação* – Receberá os design's produzidos;
- *Testes* – Realizam validações do design do sistema;
- *Gerenciamento de Projeto* – planeja o projeto e cada iteração.

3.4.2 Gerenciamento de Configurações e Alterações

No contexto deste processo Gerenciamento de Configurações e Alterações possuem como objetivos manter um conjunto consistente dos produtos dos trabalhos realizados através de sua evolução, manter *builds* consistentes do software, prover maneiras eficientes de adaptar-se a mudanças e problemas e replanejar trabalhos e, por fim, prover dados para avaliação de progresso.

Gerenciamento de Configurações está relacionado com a habilidade de manter versões dos artefatos e configurações consistentes dos artefatos relacionados com os dois primeiros objetivos apresentados. Gerenciamento de Alterações refere-se ao gerenciamento de modificações relacionadas às configurações dos artefatos apresentados nos últimos dois objetivos.

Apenas a atividade de *Request Changes* está associada a esta disciplina, sendo ela a responsável pelo gerenciamento de alterações. A disciplina está abrange todo o ciclo de vida de desenvolvimento, todas as outras disciplinas estão associadas para manter um consistente e atualizado conjunto de produtos frutos dos trabalhos realizados.

3.4.3 Implementação

Disciplina associada ao desenvolvimento das iterações do sistema cujos objetivos são verificar se as unidades técnicas utilizadas na construção do sistema funcionam como especificadas e desenvolver o produto final. Possui como atividades *Implement Developer Tests*, *Implement the Solution* e *Run Developer Tests*.

Em cada iteração as tarefas realizadas serão capazes de produzir *builds* mais estáveis, entretanto *Desenvolvedores* (tipo de ator detalhado na seção 3.5) estarão cada vez mais condicionados a arquitetura planejada.

Implementação está relacionada com outras disciplinas da seguinte maneira:

- *Requisitos* – definem o que será implementado;
- *Análise e Design* – organiza a implementação
- *Testes* – valida a implementação dos requisitos;
- *Gerenciamento de Configurações e Alterações* – gerencia as alterações realizadas nos *builds* do sistema;
- *Gerenciamento de Projeto* – planeja qual funcionalidade será implementada em cada iteração.

3.4.4 Gerenciamento de Projeto

Gerenciamento de Projeto pode ser definido como a ligação *stackholders* e a equipe de desenvolvimento. É importante que este gerenciamento desenvolva ambientes de alta produtividade onde *stackholders* tenham confiança na habilidade do time de desenvolvedores na entregar produtos condizentes a suas necessidades e que a equipe do projeto entenda essas necessidades sempre produzindo softwares de qualidade.

Esta disciplina é caracterizada por quatro tarefas (*Assess Results*, *Manage Iteration*, *Plan Iteration* e *Plan Project*) e tem como objetivos principais manter o foco da equipe no desenvolvimento e entrega para avaliação dos *stackholders*, criar um ambiente de alta produtividade, manter informados *stackholders* e equipe sobre o andamento do projeto, prover *framework* para acompanhamento dos riscos e adaptação do projeto aos mesmos e priorizar a seqüência de trabalhos.

Com todas as atividades voltadas para planejamento percebe-se a influência que o Gerenciamento de Projeto exerce nas demais disciplinas que de contraponto influenciam nas tomadas de decisões.

3.4.5 Requisitos

Pode ser definida como a disciplina responsável por obter, analisar, especificar, validar e gerenciar requisitos do sistema solicitado. Tem como principais objetivos:

- Entender o problema a ser resolvido;
- Entender as necessidades dos *stackholders*;
- Definir os requisitos da solução;
- Definir o escopo do sistema;
- Definir as interfaces externas do sistema;
- Identificar limitação técnicas da solução;
- Prover a base para o planejamento da solução;
- Prover a base inicial para estimar custo e cronograma.

Para atingir esses objetivos é importante entender os objetivos do escopo do projeto, onde os *stackholders* são os responsáveis na definição e identificação dos problemas tratados pelo produto desenvolvido. Requisitos têm três tarefas associadas que quando executadas auxiliam para completar todos os objetivos, são elas: *Define Vision*, *Detail Requirements* e *Find and Outline Requirements*.

O gerenciamento de requisitos acontece durante todo o ciclo de vida do desenvolvimento devido a mudanças requeridas pelos *stackholders*. A disciplina de Requisitos está relacionada com outras disciplinas da seguinte maneira:

- *Análise e Design* – Recebe os requisitos como seu artefato de entrada;
- *Testes* – Valida o sistema de acordo com os requisitos definidos;
- *Gerenciamento de Configurações e Alterações* – Prove os mecanismos para gerenciamento nas alterações de requisitos;
- *Gerenciamento de Projeto* – Planeja o projeto relacionando requisitos a iterações de acordo com prioridades estabelecidas para cada requisito.

3.4.6 Testes

Define o conjunto de atividades necessárias para planejamento, implementação, execução e avaliação de testes do sistema. Possui como atividades relacionadas

Create Tests Cases, *Implement Tests* e *Run Tests*, onde seus objetivos principais são encontrar e documentar defeitos, validar que o produto de software desenvolvido funciona como detalha seu design, validar e provar as propostas de design e especificação de requisitos através de demonstração e validar que a implementação dos requisitos está apropriada a definição dos mesmos.

A disciplina de Testes desafia as proposições, riscos e incertezas provenientes do trabalho das outras disciplinas utilizando demonstrações concretas e imparciais. Esta disciplina está relacionada com as demais da seguinte maneira:

- *Requisitos* – Relaciona os requisitos do projeto para estabelecer quais testes serão realizados;
- *Análise e Design* – Relaciona o design do projeto, o que ajuda na definição dos tipos de testes que serão realizados;
- *Implementação* – Disponibiliza os *builds* para efetuar os testes;
- *Gerenciamento de Projetos* – Descreve o *Iteration Plan*, este artefato define a missão correta para avaliação da iteração;
- *Gerenciamento de Configurações e Alterações* – Os testes verificam se as mudanças efetuadas foram finalizadas apropriadamente. As propriedades dos testes são gerenciadas pelo Gerenciamento de Configurações.

3.5 Atores

O OpenUP/Basic define um conjunto de sete atores principais que são alocados para a realizaçã de tarefas durante o processo de desenvolvimento, são eles: *Analista*, *Any Role*, *Arquiteto*, *Desenvolvedor*, *Gerente de Projeto*, *Stackholder* e *Tester*. Nas subseções seguintes, cada tipo de ator será definido e identificado suas principais atividades e produção de artefatos.

3.5.1 Analista

A pessoa que assume este papel tem como objetivos capturar e definir prioridades nos requisitos recolhidos a partir do entendimento dos problemas relatados pelos

stackholders. Dentre as principais habilidades necessárias para bom desempenho deste papel, podemos destacar:

- Experiência em identificar e entender problemas e oportunidades;
- Habilidade para articular necessidades associadas aos principais problemas a serem resolvidos;
- Boas habilidades de comunicação (escrita e verbal);
- Conhecimento nos domínios de negócio e tecnologia, ou habilidade de rápida absorção desses tipos de informação;

As principais tarefas alocadas para este ator são: *Define Vision*, *Detail Requirements* e *Find and Outline Requirements*. Com base nessas tarefas é responsável pelo desenvolvimento dos seguintes artefatos: *Actor*, *Glossary*, *Supporting Requirements*, *Use Case*, *Use Case Model* e *Vision*.

Figura 3.2 – Tarefas e artefatos relacionados à *Analista.*

3.5.2 Any Role

Este tipo de ator foi definido com o intuito de permitir a qualquer membro da equipe realizar tarefas gerais, como por exemplo:

- Acessar artefatos presentes no sistema de gerenciamento de configuração para desenvolvimento e manutenção dos mesmos;
- Enviar pedidos de alteração no projeto;
- Participar de revisões e definições de prioridades;
- Voluntário de trabalho em uma iteração particular.

Figura 3.3 – Tarefas relacionadas à *Any Role*.

3.5.3 Arquitetos

Este tipo de ator coordena a criação do design técnico do sistema e é responsável pelo gerenciamento de decisões de caráter técnico relacionado ao projeto. Isto significa identificar e documentar todos os aspectos arquiteturais do sistema, incluindo requisitos, design, implementação e liberação de *deploy*.

O membro de equipe alocado a esta função deve trabalhar em conjunto com o *Gerente de Projeto* (definido na subseção 3.5.5) nos aspectos de recrutamento de pessoal e planejamento do projeto, sendo responsável por reduzir riscos técnicos, gerenciar as preocupações de diferentes *stackholders* e garantir que decisões sejam repassadas, validadas e seguidas pelos membros da equipe.

Arquitetos devem ser pessoas com maturidade, visão e possuir experiência para rapidamente realizar julgamentos apesar de informações incompletas. Mais especificamente, a pessoa deve possuir a combinação de habilidades:

- Experiência nos domínios de engenharia de software e resolução de problemas;
- Habilidades de liderança;
- Excelentes habilidades de comunicação;
- Pró-ativo e seguir objetivos.

Do ponto de vista técnico devem possuir habilidades de design e implementação. De posse de todas essas qualidades, o *Arquiteto* está apto a executar as seguintes tarefas: *Analyze Architectural Requirements*, *Demonstrate the Architecture* e *Develop the Architecture*. Também será responsável pela criação e manutenção dos seguintes artefatos: *Architectural Proof-of-Concept* e *Architecture*.

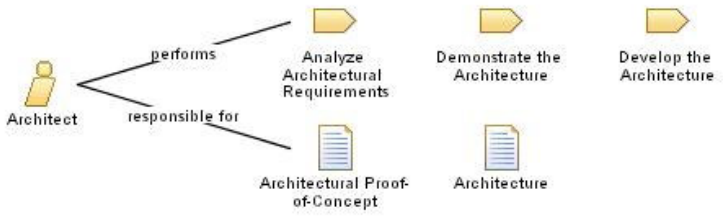


Figura 3.4 – Tarefas e artefatos relacionados ao *Arquiteto*.

3.5.4 Desenvolvedor

O membro de equipe definido neste tipo de ator é responsável pelo desenvolvimento de parte do sistema e quando necessário adaptando-a a arquitetura, possibilitando a prototipação de user-interface e depois implementando, testando e integrando partes da solução. As principais habilidades associadas são:

- Definir e desenvolver solução técnicas utilizando as tecnologias adotadas pelo projeto;
- Identificar e construir casos de teste que validem determinado comportamento de um componente do sistema;
- Descrever design's de forma que outros membros da equipe entendam.

O *Desenvolvedor* está apto a executar as seguintes tarefas: *Design the Solution*, *Implement Developer Tests*, *Implement the Solution* e *Run Developer Tests*. Também será responsável pela criação e manutenção dos seguintes artefatos: *Build*, *Design*, *Developer Test* e *Implementation*.

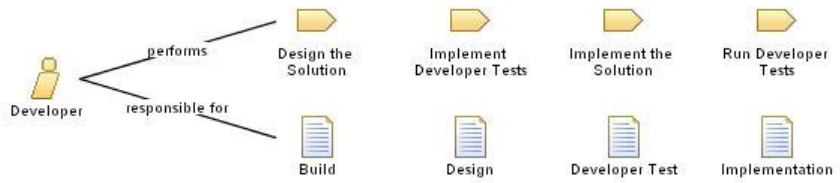


Figura 3.5 – Tarefas e artefatos relacionados ao *Desenvolvedor*.

3.5.5 Gerente de Projeto

É o responsável pelo planejamento e avaliação de riscos do projeto, coordenação de iterações com os *stackholders* e manter o foco da equipe de desenvolvimento para alcançar os objetivos do projeto. As principais habilidades necessárias são:

- Habilidade em apresentação, comunicação e negociação;
- Liderança;
- Devido a experiência com ciclos de vida de desenvolvimento de software, ser capaz de ensinar, guiar e prestar suporte a membros da equipe;
- Proficiência em resolução de conflitos e técnicas para solucionar problemas.

O *Gerente de Projeto* está apto a executar as seguintes tarefas: *Assess Results*, *Manage Iteration*, *Plan Iteration* e *Plan Project*. Também será responsável pela criação e manutenção dos seguintes artefatos: *Iteration Plan*, *Project Plan*, *Risk List*, *Status Assessments* e *Work Item List*.

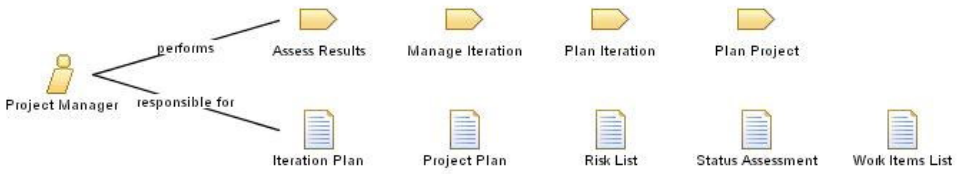


Figura 3.6 – Tarefas e artefados relacionados ao *Gerente de Projeto*.

3.5.6 Tester

Este ator é responsável pelas principais tarefas relacionadas à área de testes. Sua função é identificar os testes que deverão ser realizados, identificar o modo mais apropriado para implementar cada teste, implementar testes individuais, configurar e executar testes documentar e analisar o resultados dos testes, analisar e recuperar-se a partir de erros de execução e comunicar o resultados de testes a todos os membros da equipe.

As principais habilidades necessárias para ocupação deste cargo são:

- Conhecimento das técnicas de testes;
- Habilidades para diagnosticar e solucionar problemas;
- Conhecimento sobre o sistema que está sendo testado;
- Conhecimento sobre a arquitetura de sistemas e redes.

Caso testes automatizados sejam utilizados, as seguintes qualidades deverão ser consideradas para alocação de membros nesta função:

- Habilidades em diagnóstico e *debugging*;
- Habilidades em programação;
- Experiência utilizando ferramentas de automação de testes;
- Treinamento apropriado para utilização da ferramenta de automação de testes em questão.

O *Tester* está apto a executar as seguintes tarefas: *Create Test Cases*, *Implement Tests* e *Run Tests*. Também será responsável pela criação e manutenção dos seguintes artefatos: *Test Case*, *Test Log* e *Test Script*.

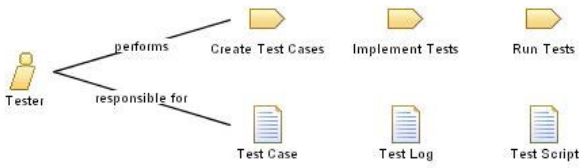


Figura 3.7 – Tarefas e artefatos relacionados ao *Tester*.

3.5.7 Stackholder

Representa o grupo de pessoas cujo conjunto de necessidades deve ser satisfeito ao final do projeto. Este ator pode ser atribuído a qualquer pessoa que será (ou potencialmente) afetado pelo resultado do projeto.

3.6 Atividades

As Atividades definem o fluxo de tarefas e produção e utilização de artefatos por parte delas, devendo ser realizadas pelas iterações em suas respectivas fases do ciclo de desenvolvimento. Abaixo segue a lista de todas as atividades definidas pelo OpenUP/Basic.

Initiate Project

É realizada no começo da fase de *Inception* pela primeira iteração. Tem como objetivos principais estabelecer a visão do projeto e estabelecer o planejamento do mesmo em um maior grau de abstração.

Define Architecture

Esta atividade tem como propósito a definição da arquitetura que possibilita a implementação dos requisitos de maior prioridade, provendo uma fundação sólida e resistente onde serão desenvolvidas as funcionalidades. Será realizada por todas as iterações da fase de *Elaboration*.

Develop Solution (for requirements) (within context)

Design, implementar, testar e integrar a solução para um requisito em um determinado contexto. O contexto é especificado ao alocar o requisito para o desenvolvedor. O foco de desenvolvimento pode ser em uma camada (apresentação, negócios, persistência, etc.), um componente, etc.

Determine Architectural Feasibility

Tendo como base os requisitos do projeto, o arquiteto deverá propor uma arquitetura de alto nível condizente com as funcionalidades e restrições do sistema, maximizando os benefícios aos *stackholders*. Realizada durante a fase de *Inception*.

Manage Iteration

Desempenhada unicamente pelo gerente de projeto, tem como objetivo gerenciar todas as atividades alocadas em cada iteração, monitorando e comunicando o status do projeto aos *stackholders*. Esta atividade está presente em todas as fases do ciclo de desenvolvimento.

Manage Requirements

Esta atividade ocorre durante todo o ciclo de vida de desenvolvimento através da colaboração de todos os membros da equipe e *stackholders* a fim de estabelecer o conjunto consistente, correto e passível de validação de requisitos do sistema. De modo geral, suas tarefas são obter, especificar, analisar e validar esse conjunto de requisitos.

Ongoing Tasks

Atividade criada com finalidade de "abrigar" a tarefa de *Request Changes*. Esta tarefa é executada por qualquer membro do sistema em qualquer fase do ciclo de desenvolvimento.

Validate Build

Criar e executar testes capazes de validar o desenvolvimento de funcionalidades condizentes com os requisitos associados ao sistema.

3.7 Fases do Ciclo de Desenvolvimento

O OpenUP/Basic é um processo iterativo onde as iterações são distribuídas ao longo de quatro fases: *Inception*, *Elaboration*, *Construction* e *Transition*. A quantidade existente de iterações por fase dependerá do domínio de negócios, tecnologia utilizada, complexidade da arquitetura, tamanho do projeto, entre outros. Cada fase define um conjunto de objetivos que devem ser alcançados por suas iterações, assim como a associação com as atividades (seção 3.6) realizadas para atingir esses objetivos.

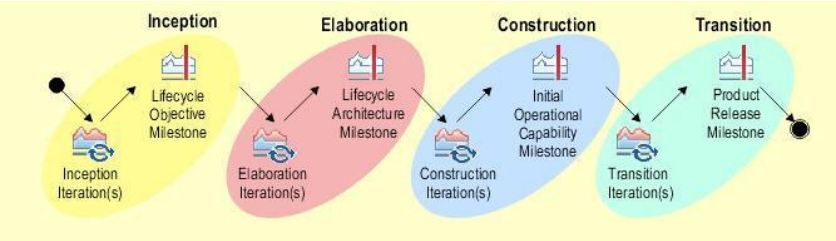


Figura 3.8 – Ciclo de vida do OpenUP/Basic.

Iterações podem ter durações variáveis dependendo das características do projeto, entretanto, é recomendado o período de um mês por iteração, pois este período prove:

- Quantidade de tempo razoável para entrega de novas funcionalidades;
- Freqüentes *feedbacks* de *stackholders*;
- Gerenciamento de riscos e problemas durante o decorrer do projeto.

3.7.1 Inception Phase

A primeira das quatro fases do ciclo de vida do processo está relacionada com o entendimento sobre o escopo do projeto e seus objetivos, buscando informações que indiquem a viabilidade do mesmo. O propósito desta fase é estabelecer concordância com todos os *stackholders* sobre os objetivos do ciclo de vida do projeto.

Para entender o escopo do projeto, assim como a viabilidade de uma possível solução, quatro objetivos devem ser contemplados, são eles:

- Entender o que deve ser construído;
- Identificar funcionalidades chave do sistema;

- Determinar pelo menos uma solução;
- Entender custos, cronograma e riscos associados ao projeto.

Para atingir os objetivos traçados acima, esta fase está associada às seguintes atividades:

- *Initiate Project* – Resulta na criação da visão e do planejamento do projeto;
- *Manage Iteration* – Busca do entendimento das necessidades dos *stackholders* resultando nos requisitos do sistema, lista dos riscos e custos e planejamento do cronograma do projeto.
- *Manage Requirements* – Focada no mapeamento das necessidades dos *stackholders* em requisitos, estabelecendo suas prioridades e documentando-os em casos de uso. Possibilita ao arquiteto elaborar um protótipo da arquitetura, entregando-a ao gerente do projeto a fim de aprimorar o planejamento do projeto e de futuras iterações.
- *Determine Architectural Feasibility* – Realizada pelo arquiteto, são criados protótipos de algumas possíveis arquiteturas para o sistema, determinando a viabilidade do projeto e por fim escolhendo a arquitetura a ser implementada.

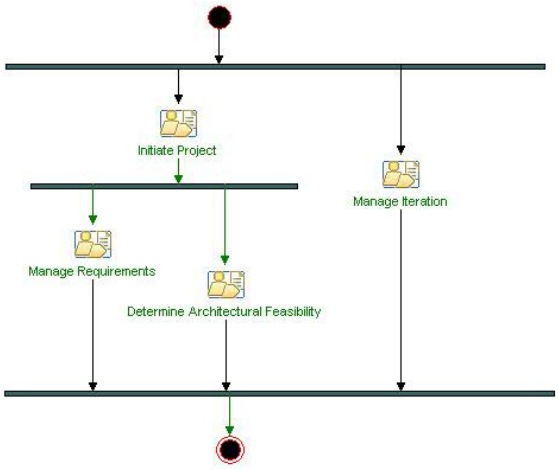


Figura 3.9 – Fluxo de atividades da fase *Inception*.

A tabela a seguir demonstra o relacionamento entre atividades e objetivos da fase de *Inception*:

Objetivo	Atividade(s)
Entender o que deve ser feito	-Manage Requirements
Identificar funcionalidades chave do sistema	-Initiate Project -Manage Requirements
Determinar pelo menos uma solução	-Determine Architectural Feasibility
Entender custos, cronograma e riscos associados ao projeto	-Initiate Project -Manage Iteration

Tabela 2 – Fase *Inception*: Objetivos x Atividades.

3.7.2 Elaboration Phase

A segunda das quatro fases do ciclo de vida do processo está focada nos riscos relacionados à arquitetura do sistema. O propósito desta fase é estabelecer um *baseline* de criação da arquitetura do sistema, capaz de suportar todas as atividades de desenvolvimento seguintes.

Os objetivos desta fase contemplam o gerenciamento de riscos atrelados aos requisitos, a arquitetura, custo e cronograma do projeto. Os objetivos são:

- Entendimento mais detalhado dos requisitos;
- Design, implementar, validar e estabelecer *baselines* da arquitetura;
- Tratar prioridades e produzir cronogramas e estimativas de custos de forma mais precisa.

Para atingir os objetivos traçados acima, esta fase define quatro atividades:

- *Manage Iteration* – O gerente de projeto, em conjunto com os membros da equipe de desenvolvimento, analisa a lista de trabalho da iteração gerando uma estimativa de prazos mais precisa. As tarefas são alocadas entre os desenvolvedores que passam a implementar as funcionalidades definidas por requisitos de alta prioridade. Também são extraídas métricas para auxiliar estimativas de cronograma e custo de futuras iterações;
- *Manage Requirements* – Todos os requisitos de alta prioridade são destrinchados e detalhados ao ponto de possibilitar a implementação de suas funcionalidades. Novos requisitos poderão ser relacionados necessitando análise;

- *Define the Architecture* – O arquiteto analisa as restrições associadas à arquitetura e lista as prioridades no desenvolvimento do sistema, definindo a estrutura do projeto e identificando as abstrações iniciais e mecanismos disponibilizados pela arquitetura;
- *Develop Solution (for requirements) (within context)* – A equipe de desenvolvimento recebe um subconjunto dos requisitos do sistema para implementação de funcionalidades associadas. Durante esta fase, a maioria dos requisitos tratados pela atividade possui alta prioridade;
- *Validate Build* – Nesta fase, esta atividade tem como propósito testar e validar a implementação dos requisitos de alta prioridade, provando a robustez da arquitetura.

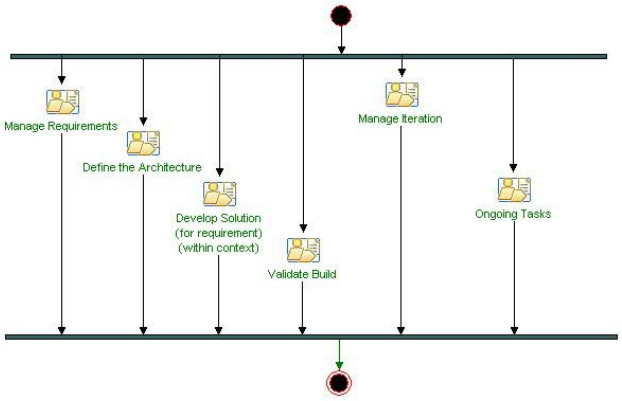


Figura 3.10 - Fluxo de atividades da fase *Elaboration*.

A tabela a seguir demonstra o relacionamento entre atividades e objetivos da fase de *Elaboration*:

Objetivo	Atividade(s)
Entendimento mais detalhado dos requisitos	-Manage Requirements
Design, implementar, validar e estabelecer *baselines* da arquitetura	-Define the Architecture -Develop Solution (for requirements) (within context) -Validate Build
Minimizar riscos e produzir cronogramas e estimar custos de forma precisa	-Manage Iteration

Tabela 3 – Fase *Elaboration*: Objetivos x Atividades.

3.7.3 Construction Phase

A terceira das quatro fases do ciclo de vida do processo possui sua estrutura de trabalhos similar a *Elaboration Phase*, onde suas atividades são desempenhadas em paralelo. Ao iniciar esta fase a arquitetura deverá estar consolidada restando apenas requisitos de baixa prioridade para implementação. Funcionalidades serão testadas e validadas resultando em *builds* mais completos e estáveis. Assim, permite ao gerente do projeto foco na redução de custos do projeto e gerenciamento e análise da eficiência da equipe.

Dois objetivos devem ser contemplados, são eles:

- Desenvolver produtos completos que possam ser liberados à comunidade;
- Minimizar custos de desenvolvimento e atingir paralelismo no desenvolvimento.

Para atingir os objetivos traçados acima, esta fase define quatro atividades:

- *Manage Iteration* – Atividades focadas na liberação de uma "versão *beta*" estável do sistema aos *stackholders*;
- *Manage Requirements* – Requisitos de baixa prioridade são detalahados e repassados a equipe de desenvolvimento;
- *Develop Solution (for requirements) (within context)* – Foco na implementação de requisitos de médio/baixa prioridade;
- *Validate Build* – Validar e liberar a "versão *beta*" disponibilizada ao final da iteração.

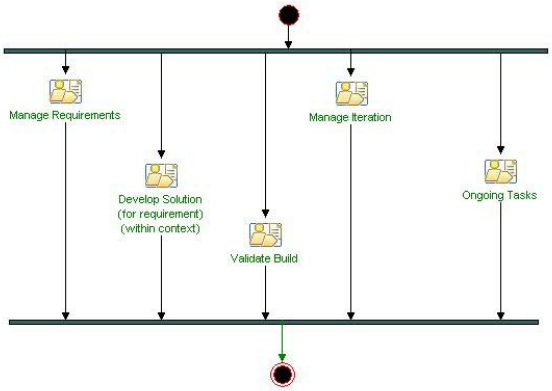


Figura 3.11 - Fluxo de atividades da fase *Construction*.

A tabela a seguir demonstra o relacionamento entre atividades e objetivos da fase de *Construction*:

Objetivo	**Atividade(s)**
Desenvolver produtos completos que possam ser liberados à comunidade	-Manage Requirements -Develop Solution (for requirements) (within context) -Validate Build
Minimizar custos de desenvolvimento e atingir paralelismo no desenvolvimento	-Manage Iteration -Develop Solution (for requirements) (within context) -Validate Build

Tabela 4 – Fase *Construction*: Objetivos x Atividades.

3.7.4 Transition Phase

A última das quatro fases do ciclo de vida do processo tem o propósito de assegurar que todas as funcionalidades serão entregues na liberação do produto.

Possui três objetivos que devem ser contemplados, são eles:

- Avaliar, através de testes, se as expectativas dos usuários serão atendidas;
- Stockholders estabelecem que o deploy está completo;
- Melhorar performance de futuros projetos baseados nas lições aprendidas.

Para atingir os objetivos traçados acima, esta fase define quatro atividades:

- *Manage Iteration* – As atividades devem convergir para liberação de uma versão estável do software, permitindo a aprovação por parte dos *stackholders*;
- *Develop Solution (for requirements) (within context)* – Durante esta fase a maioria dos requisitos estarão implementados e validados, entretanto alguns requisitos de baixa prioridade podem ser desenvolvidos e incluídos na liberação da versão. Alguns erros apresentados em outras iterações deverão ser assistidos e validados antes do *deploy*;
- *Validate Build* – Efetuada em paralelo com as demais, está atividade verifica a implementação dos requisitos, atestando a estabilidade da versão liberada para o conjunto de usuários.

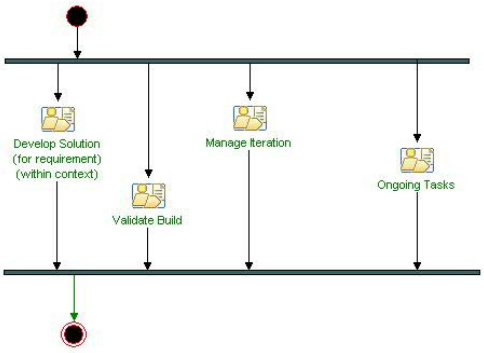


Figura 3.12 – Fluxo de atividades da fase *Transition*.

A tabela a seguir demonstra o relacionamento entre atividades e objetivos da fase de *Transition*:

Objetivo	Atividade(s)
Avaliar, através de testes, se as expectativas dos usuários serão atendidas	-Ongoing Tasks -Develop Solution (for requirements) (within context) -Validate Build
Stockholders estabelecem que deploy está completo.	-Manage Iteration -Validate Build
Melhorar performance de futuros projetos baseados nas lições aprendidas	-Manage Iteration

Tabela 5 – Fase *Transition*: Objetivos x Atividades.

3.8 Considerações Finais

Este capítulo apresentou os principais conceitos relativos ao OpenUP/Basic. Foi dada uma explicação sobre sua estrutura através da descrição de seus princípios, áreas de conteúdo (ou simplesmente subprocessos), disciplinas e também demonstrado aspectos importantes de seu ciclo de desenvolvimento, tais como atores, atividades e fases de iterações. A descrição dada neste capítulo servirá de base para apresentação do trabalho de extensão mostrado no próximo capítulo.

Capítulo 4

Estendendo o OpenUP/Basic

De acordo com [15,21] muito são os aspectos que diferenciam os aplicativos Web dos aplicativos tradicionais, dentre os quais podemos citar navegação e organização de interface. O OpenUP/Basic torna-se o processo de desenvolvimento perfeito a alcançar os objetivos deste trabalho devido a sua capacidade de absorver extensões sem perder suas características originais.

Neste capítulo são apresentadas as extensões idealizadas para as atividades de Requisitos e Analise e Projeto.

4.1 Visão geral

Nos capítulos anteriores apresentamos as maravilhas trazidas pela Web, como a troca de informações de maneira mais rápida e eficiente, contudo listamos alguns pontos fracos dos processos de desenvolvimento de software utilizados para construir aplicativos Web, visto que estes não estão completamente preparados para enfrentar os problemas relacionados a este ambiente tão heterogêneo.

O OpenUP/Basic é um processo de desenvolvimento iterativo baseado nas metodologias RUP e XP, onde uma de suas principais características é ser passível de extensão. Tendo em vista esta capacidade, o OpenUP/Basic foi selecionado com o objetivo de estender suas atividades relacionadas às disciplinas de Requisitos e Análise e Projeto para melhor adequar-se as características dos aplicativos Web.

No capítulo 3, o OpenUP/Basic foi extensivamente detalhado, portanto neste capítulo abordaremos apenas os aspectos relacionados com o trabalho de extensão.

4.2 Estendendo a disciplina de Requisitos

A Engenharia de software define requisitos como o conjunto das funcionalidades e restrições [49] apresentadas pelo sistema. A elicitação de requisitos é uma atividade importante no processo de desenvolvimento de software por estabelecer, dentre várias coisas, a ordem na implementação das necessidades apontadas pelos *stackholders*. Requisitos podem ser divididos em três grupos:

- Requisitos funcionais – declarações de funções que o sistema deve fornecer, especificando o comportamento, entradas e saídas dessas funções;
- Requisitos não-funcionais – São restrições sobre as funções do sistema;
- Requisitos de domínio – Restrições originadas do domínio em que a aplicação estará submetida.

No OpenUP/Basic a disciplina de Requisitos está associada exclusivamente a atividade de *Manage Requirements* (Figura 4.1 demonstra seu fluxo de tarefas)

definindo um conjunto de tarefas responsável por obter, especificar, analisar e validar o conjunto de requisitos do sistema.

Submetendo aplicativos Web e tradicionais as tarefas desta atividade, notamos que não existem grandes diferença no gerenciamento dos requisitos funcionais. Mas é perceptível que nos aplicativos Web um conjunto de requisitos não-funcionais deve ser obrigatoriamente tratado independente da finalidade da aplicação.

Abaixo são listados esses requisitos essenciais, definindo-os e apresentando possíveis impactos quando mal trabalhados pelas equipes de projeto:

- *Concorrência* – Sistemas Web tendem a possuir grandes números de requisições, portanto tanto a tecnologia utilizada para implementação da solução quanto a estrutura de física deve ser capaz de suportar requisições concorrentes de funcionalidades. Quando este aspecto não é bem trabalhado entre equipes e stackholders são grandes as chances da solução ser inviabilizada, pois servidores e aplicações não agüentaram a demanda dos usuários;
- *Persistência* – De informações e consistência nas funcionalidades, podendo acorrer desentendimentos entre prestadores do serviço e usuários devido a informações mal trabalhadas pelo sistema;
- *Multi-plataforma* – Sistemas Web geralmente trabalham em ambientes heterogêneos não havendo conformidade de hardware ou software. É importante verificar quais as características presentes na estrutura que alojará a aplicações para que incompatibilidades de tecnologias não resultem em percalços ou até o fracasso do projeto;
- *Cross-browser* – Apesar do controle exercido pela W3C [50], diferentes companhias implementam as *tags* da linguagem de hipertexto (html) de maneiras diferentes em seus respectivos *browsers*, o que pode acarretar em experiências de interface completamente distintas entre usuários que não utilizam o mesmo tipo de *browsers*. Gerentes de projeto, arquitetos, web designers e *stackholders* devem estar cientes desses problemas e buscar soluções conjuntas para contorná-los;

- *Segurança* – Um dos principais aspectos relacionados a aplicativos Web. Compreende desde acesso e transmissão de informações sigilosas até permissão para execução de funcionalidades específicas. Tratamento indevido por parte dos sistemas pode acarretar em processos judiciais;
- *Bandwidth* – Julga a capacidade de receber e transmitir dados do servidor do cliente. Cálculos errados podem decretar morte do sistema viso que usuários não receberão as informações requisitadas. Este aspecto aumenta de sua importância de acordo com o aumento de usuários utilizando concorrentemente;
- *Disponibilidade* – É uma das vantagens associadas a internet que deve ser aproveitada ao máximo pelos sistemas Web. A exposição de produtos e serviços durante as 24 horas diárias torna-se um diferencial competitivo, permitindo aos clientes usufruir dos benefícios de uma empresa ou órgão governamental para realização de todas as suas atividades. A retirada de serviços do ar pode gerar graves problemas aos usuários impedidos de finalizar suas tarefas;
- *Escalabilidade* – "O Céu é o limite". Sistemas Web devem permitir fácil manutenção e incorporação de novas funcionalidades. O risco está associado a necessidade de desenvolver uma nova aplicação sempre que for necessária a adição de novas funcionalidades ou retirada de acesso de todos os serviço sempre que uma nova versão do sistema seja liberada;
- *Multiplicidade de perfis* – Tratamento diferenciado aos diversos grupos de usuários, definindo acesso a funcionalidades e informações além de possíveis personalizações da interface do sistema. Pessoas desejam tratamento diferenciado e apenas as soluções capazes de lhes proporcionar sobreviverão.
- *Dinâmica de conteúdo* – Na era da internet o ciclo de vida de informação é muito curto, visto que a todo instante "surge uma novidade". Provedores de serviços devem estar sempre atentos ao repasse das novas informações aos usuários. Estes estão na eterna busca por produtos e serviços novos capazes de proporcionar melhor qualidade de vida.

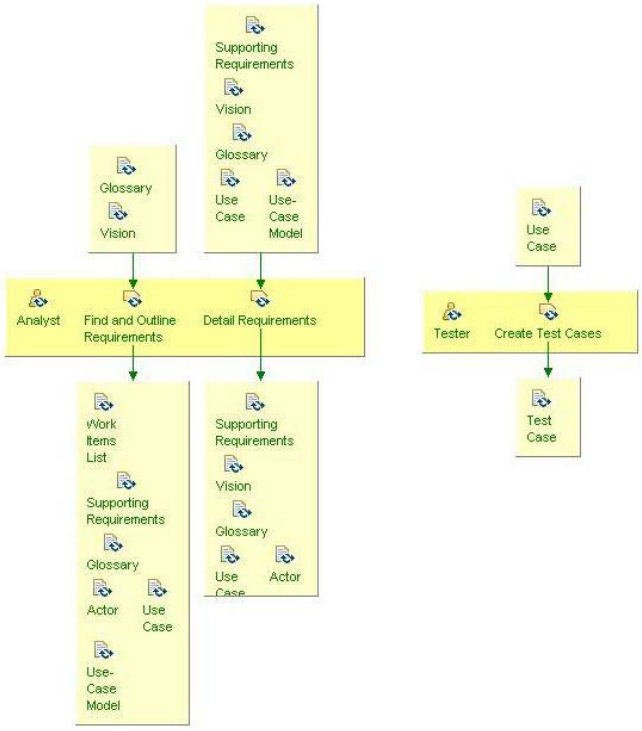


Figura 4.1 – Fluxo de tarefas de *Manage Requirements*.

4.3 Extensão das atividades de Análise e Projeto

A disciplina de Análise e Projeto está baseada em duas atividades: *Determine Architectural Feasibility* e *Define Architecture*. A primeira tem início na fase de *Inception* e termina na fase de *Elaboration* e possui o objetivo de estabelecer uma possível solução para a arquitetura do sistema e provar a sua viabilidade. Já a segunda é completamente realizada durante a fase de *Elaboration*, com propósito definir estrutura capaz de comportar os requisitos de alta prioridade de cada iteração.

Para que as tarefas dessas atividades estejam completamente aptas ao desenvolvimento de aplicações Web faz-se necessário um artefato que defina a estrutura, o comportamento e navegabilidade das interfaces do usuário. Devido à importância atribuída à camada de apresentação por parte dos sistemas Web, as equipes de projetos devem possuir um profissional ciente das técnicas de usabilidade capaz de criar o modelo navegacional da aplicação (seção 4.3.1.2). Este novo tipo de ator será chamado de *Web Designer*.

4.3.1 Web Designer

Este membro da equipe do projeto tem como objetivo de desenvolver o projeto navegacional do aplicativo Web a partir dos requisitos relatados. Dentre as principais habilidades necessárias para bom desempenho deste papel, podemos destacar:

- Conhecimento de técnicas de usabilidade;
- Habilidades de comunicação (escrita e verbal);
- Pró-atividade;
- Domínio de tecnologias de desenvolvimento para Web.

Este ator é responsável pela tarefa: *Create Interface Design*, sendo necessário desenvolver o artefato *Navigational Model* ao final da tarefa e distribuir entre todos os membros da equipe.

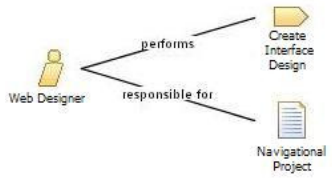


Figura 4.2 Tarefa e artefato relacionados ao Web Designer.

4.3.1.1 Create Interface Design

Definição: Definir o espaço navegacional da aplicação a partir dos requisitos e restrições impostas ao sistema.

Ator Responsável: Web Designer.

Artefatos de Entrada:

- Supporting Requirements;
- Vision;
- Use Case Model.

Artefato de Saída: Navigational Model.

Passos:

- *Analisar casos de uso* – Retirar informações relacionadas à estrutura da futura interface, tais como tipos de conteúdos (texto, vídeos, imagens) visualizados, abordagens para diferentes tipos de usuários disponíveis no sistema, etc.;
- *Listar funcionalidades acessíveis ao cliente* – Verificar quais funcionalidades serão disponibilizadas aos usuários e definir a posição da chamada dessas funcionalidades no espaço navegacional;
- *Listar conteúdos acessíveis ao cliente* – Verificar quais informações serão disponibilizadas aos usuários e definir a forma e lugar onde serão exibidas;
- *Listar navegação entre interfaces e outros sistemas* – Definir os links entre as diversas interfaces do sistema e o relacionamento com outros sistemas e Web sites;
- *Construir Modelo Navegacional* – De posse das informações geradas nos passos anteriores o modelo navegacional é construído;
- *Validar e distribuir modelo* – Neste passo o modelo é revisado e entregue aos demais membros da equipe.

4.3.1.2 Navigational Model

Definição: Conjunto de protótipos das interfaces do sistema, contendo informações sobre visualização de funcionalidades e conteúdos por partes dos diferentes perfis de usuários, bem como informações sobre navegação entre interfaces e com outros sistemas.

Responsável: Web Designer.

Entrada de:

- Demonstrate the Architecture;
- Analyze Architectural Requirements;
- Develop the Architecture.

Saída de:

- Create Interface Design.

4.3.2 Adaptação das tarefas

Com os novos recursos apresentados definido, algumas modificações foram sugeridas nas atividades de Análise e Projeto. Para a atividade *Determine Architecture Feasibility* adicionamos ao fluxo da tarefa desempenhada pelo Web Designer, *Create Interface Design*, onde o artefato de saída dessa tarefa, *Navigational Model*, será utilizado como artefato de entrada nas tarefas do Arquiteto, *Analyze Architectural Requirements* e *Demonstrate the Architecture*, com a finalidade de prover uma base mais sólida para as soluções propostas e planejamento mais adequado ao desenvolvimento de aplicativos Web.

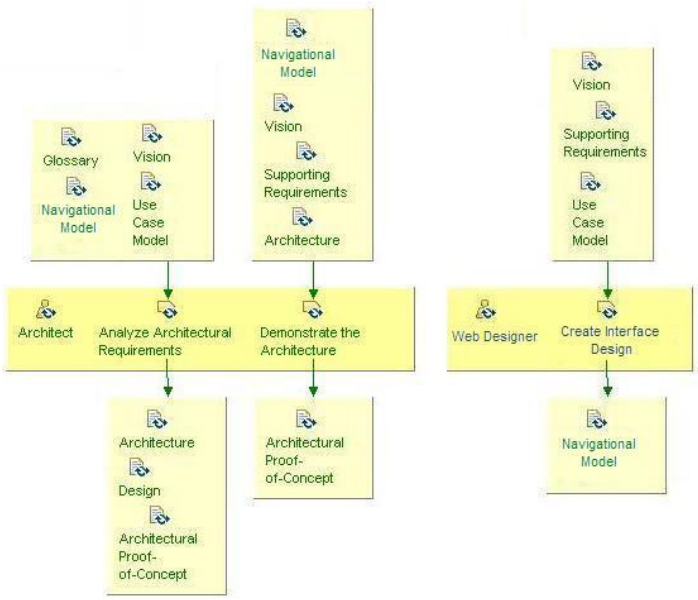


Figura 4.3 – Novo fluxo da atividade *Demonstrate Architectural Feasibility*.

Com relação ao fluxo da atividade de *Define Architecture*, não são necessárias grandes alterações (inclusão de novas tarefas, participação de outros tipos de atores ou mudanças nos fluxos internos das tarefas), mas para munir as tarefas de melhores informações o artefato produzido na tarefa de *Create Interface Design* será incorporado ao conjunto de documentos de entrada de cada uma delas, *Analyze Architecture Requirements* e *Develop Architecture*.

A atribuição de um novo artefato de entrada às tarefas desta atividade tem como objetivo ratificar a importância dos aspectos de interface no desenvolvimento de um aplicativo Web e munir o planejamento das iterações, esta atividade é executada na fase de *Elaboration*, de maneira que estas sejam separadas por módulos com foco em grupos de funcionalidades.

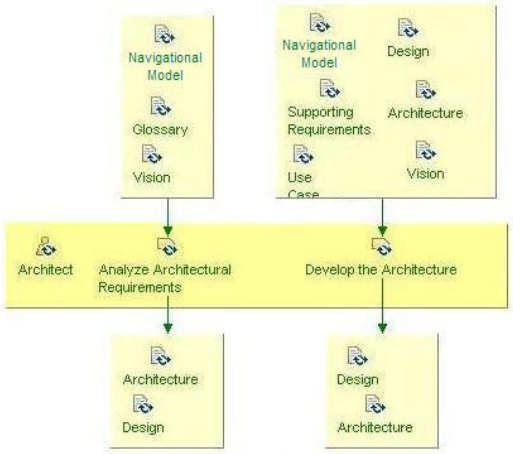


Figura 4.4 – Novo fluxo da atividade *Define Architecture*.

4.4 Considerações Finais

Este capítulo apresentou todas as propostas de extensão relativas às atividades de Requisitos e Análise e Projeto do OpenUP/Basic. Foram detalhadas as atividades originais indicando o papel das extensões na realização das mesmas, contribuindo para adaptação do processo de desenvolvimento e elaboração de aplicativos Web de alta qualidade.

No próximo capítulo serão apresentadas as conclusões, considerações finais, trabalhos correlatos e as perspectivas de trabalhos futuros.

Capítulo 5

Conclusão

Com a disseminação da Web perante todos os seguimentos da sociedade e aumento na utilização de seus serviços, existe uma crescente demanda por aplicativos Web de alta complexidade. Entretanto, este tipo de aplicações possui características que não são abordadas pelos processos de desenvolvimento tradicionais.

Diante deste cenário, este trabalho esteve focado no tratamento das características especiais dos aplicativos Web. Para Tanto, desenvolveu uma extensão do processo de desenvolvimento OpenUP/Basic, com foco nas atividades relacionas a Requisitos e Análise e Projeto de Sistema.

Este capítulo apresenta as considerações gerais e contribuições, além de trabalhos correlatos e perspectivas de trabalhos futuros.

5.1 Considerações Finais e contribuições

A Web tornou-se a maior fonte para publicação e extração de informação na era da Informação. De certo modo, a Web quebrou barreiras na disseminação de informações permitindo que todos tenham acesso a elas seja independentemente de onde esteja e como estejam conectados.

Por ser um ambiente de grande alcance e rápida expansão [14], torna-se perfeita para investimentos pesados por parte de empresas com o objetivo de expor seus produtos e serviços a clientes de todas as partes do mundo [31,32,33], bem como propiciar aos funcionários informações valiosas para realização de seus trabalhos. Não são apenas empresas que investem no mundo virtual, muitos órgãos governamentais estão migrando seus serviços para a grande rede de computadores.

Diante dessa perspectiva, os aplicativos Web estão sendo produzidos com maior riqueza e complexidade, entretanto, por possuírem características diferenciadas as dos aplicativos tradicionais, a maioria dos processos de software não se encontra preparada para tratar essas diferenças.

Este trabalho teve como objetivo melhor adaptar um processo de desenvolvimento a realidade do ambiente e ciclo de vida dos aplicativos Web, para tanto foi escolhido o processo OpenUP/Basic. O OpenUP/Basic é um processo de desenvolvimento de software criado pela IBM [29] e doado à Eclipse Foundation [28], sendo o refinamento dos principais aspectos do *Rational Unified Process* (RUP) [25] e *Extreme Programming* (XP) [24].

Devido a sua simplicidade, por combinar aspectos importantes de ambas as metodologias, RUP e XP, e estar focado no desenvolvimento de aplicativos de pequeno/médio porte com duração de três a seis meses, o OpenUP/Basic foi escolhido.

As atividades de Requisitos e Análise e Projeto foram estendidas para atender características importantes dos aplicativos Web. Em virtude de suas propriedades, o

processo é passível de extensões, nenhuma das propostas deste trabalho prejudica a dinâmica deste processo, permitindo que continue sendo utilizado pelos tipos de aplicações.

A análise e gerenciamento de requisitos definem dois tipos de requisitos: Requisitos funcionais e requisitos não-funcionais. Percebemos que não existe diferença nas técnicas ou precauções que devam ser tomadas quanto ao gerenciamento de requisitos funcionais por parte das aplicações Web e aplicativos tradicionais. Entretanto, devemos compreender que os requisitos não-funcionais têm grande influência no planejamento e validação de um projeto Web.

Com base nessas considerações, a extensão das atividades de Requisitos enumera os principais aspectos das aplicações Web, explicitando o impacto desses aspectosnos aplicativos, que obrigatoriamente devem ser tratados em prol da viabilização.

A extensão das atividades de Análise e Projeto está voltada para o desenvolvimento da camada de Interface consequentemente dando base à definição completa da arquitetura do sistema. Para tanto, exige estudo de navegabilidade da aplicação em questão, baseando-se na multiplicidade de perfis dos usuários e acesso a funcionalidades e conteúdos por parte desses usuários.

Portanto, acreditamos que a extensão do OpenUP/Basci atingiu seus objetivos de prover ao desenvolvimento de aplicações Web um modelo adaptado a suas principais características e exigências.

5.2 Trabalhos relacionados

Alguns trabalhos produzidos possuíam o mesmo enfoque apontado por este, abordando metodologias de desenvolvimento diferentes, entre eles temos:

- XWebProcess – Extensão das disciplinas de *Extreme Programming* (XP) com o objetivo de melhor adapta-las a realidade do ciclo de vida do desenvolvimento de aplicativos Web [16].

- Extensão RUP – Baseada no *Rational Unified Process* (RUP), refinou os fluxos de Requisitos e Análise e Projeto do RUP para atender as características dos aplicativos Web [17].

Nosso trabalhou adaptou o processo de desenvolvimento OpenUP/Basic, pois sua abordagem é mais interessante sendo baseada nos dois processos acima (XP, RUP). Outro aspecto positivo em favor do OpenUP/Basic é sua adaptação a características na formação das equipes de projeto (média de 3 a 6 membros) e duração dos projetos (média de 3 a 6 meses).

5.3 Trabalhos futuros

O OpenUP/Basic é um processo de desenvolvimento de software baseado nos processos RUP e XP. A extensão de atividades voltadas para requisitos e análise e projeto (essencialmente relacionadas com as disciplinas de Análise e Design e Requisitos) tiveram como foco adaptá-lo ao desenvolvimento de aplicativos Web.

Como perspectiva de trabalho futuros seria interessante a validação das alterações das atividades através de um estudo de caso e adaptação de outras disciplinas e atividades para o desenvolvimento de aplicações Web, bem como o desenvolvimento de ferramentas, adaptadas ao processo, permitindo que métricas utilizadas nas estimativas de custo e tempo sejam mais apropriadamente calculadas, otimizando o trabalho dos gerentes de projeto e beneficiando o trabalho das equipes de desenvolvimento como um todo.

Outra perspectiva interessante seria comparar os diversos processos verificando seus pontos positivos e negativos, refinando-os para melhor adaptação a realidade Web.

Referências bibliográficas

[1] Sun Microsystem. *Java*. Disponível em: http://java.sun.com. Último acesso em 14/03/2007.

[2] W3C. *Request-reponse paradigm*. Disponível em: http://www.w3c.com. Último acesso em 14/03/2007.

[3] Apache Foundation. *Struts*. Disponível em: http://struts.apache.org. Último acesso em 14/03/2007.

[4] Red Hat. JBoss Enterprise Middleware Suíte. *Hibernate*. Disponível em: http://www.hibernate.org. Último acesso em 14/03/2007.

[5] Apache Foundation. *Tomcat*. Disponível em: http://tomcat.apache.org. Último acesso em 14/03/2007.

[6] Apache Foundation. Disponível em: http://www.apache.org, Último acesso em 14/03/2007.

[7] Tanenbaum, A.S. Computer Networks, Prentice Hall, 1996.

[8] Wikipedia. *ARPANET*. Disponível em: http://en.wikipedia.org/wiki/ARPANET. Último acesso em: 15/03/2007.

[9] Wikipedia. *Internet*. Disponível em: http://en.wikipedia.org/wiki/Internet. Último acesso em: 15/03/2007.

[10] Berners-Lee, Tim, et al. *Architecture of the World Wide Web, Volume One*. W3C, disponível em: http://www.w3.org/TR/webarch/. Último Acesso: 15/03/2007.

[11] Azevedo, W. *Poética das Hipermídias*. São Paulo, editora Mackenzie, disponível em: http://www.mackenzie.com.br/interacao/www2003/poeticadaship.pdf. Último acesso em 15/03/2007.

[12] Leão, L. O Labirinto da Hipermídia. São Paulo, Iluminuras, 1999.

[13] Parente, A. O Virtual e o Hipertextual. Rio de Janeiro: Pazulin, 1999.

[14] Internet World Stats. Disponível em: http://www.internetworldstats.com. Último acesso em 15/03/2007.

[15] Rossi, G., Schwabe, D., Lyardet, F. *Web application models are more than conceptual models*. First International Workshop on Conceptual Modeling and the WWW, 1999.

[16] Sampaio, A. T. F. *XWEBPROCESS: Um processo ágil para desenvolvimento WEB*. Dissertação de mestrado, Centro de Informática - Universidade Federal de Pernambuco, 2004.

[17] Souza, R. A. C. de *Uma Extensão do Fluxo de Análise e Projeto do RUP para o Desenvolvimento de Aplicações WEB*. Dissertação de Mestrado, Centro de Informática – Universidade Federal de Pernambuco, 2002.

[18] Marques, H. M. *Um Processo para Customização de Produtos de Software*. Dissertação de Mestrado, Centro de Informática – Universidade Federal de Pernambuco, 2005.

[19] Didier, A. C. V. B. *WRE-Process: Um processo de Engenharia de Requisitos Baseado no RUP*. Dissertação de Mestrado, Centro de Informática – Universidade Federal de Pernambuco, 2003.

[20] Reis, T. P. C. *REQE – Uma Metodologia para Medição de Qualidade de Aplicações WEB na Fase de Requisitos*. Dissertação de Mestrado, Centro de Informática – Universidade Federal de Pernambuco, 2004.

[21] Schwabe, D., Rossi, G. *An Object Oriented Approach to Web-Based Applications Design*. TAPOS - Theory and Practice of Object Systems, vol. 4, pp. 207-225, 1998.

[22] Cardoso, R. F. *AvalWeb – Sistema interativo para gerência de questões e aplicação de avaliações Web*. Dissertação de Mestrado, Instituto de Informática – Universidade Federal do Rio Grande do Sul, 2001.

[23] Nogueira, G., Capra, S. *Adaptação do RUP para Projetos de Software e-Commerce*. Trabalho de Conclusão de Curso, Centro de Ciências Exatas e da Natureza – Universidade da Amazônia, 2003.

[24] Beck, K. *Extreme Programming Explained*, Addison-Wesley, 1999.

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

[26] Booch, G., Rumbaugh, J., Jacobson, I. *The Unified Modeling Language: User Guide*. Addison-Wesley, 1999.

[27] Eclipse Foundation. Disponível em: http://www.eclipse.org. Último acesso em: 17/03/2007.

[28] Eclipse Foundation. *Eclipse Process Framework (EPF)*. Disponível em: http://www.eclipse.org/epf. Último acesso em: 24/03/2007.

[29] IBM. Disponível em: http://www.ibm.com. Último acesso em: 17/03/2007.

[30] Hech, E., Vervest, P. *How should CIOs deal with Web-based Auctions*, ACM, 1998.

[31] Braun, D. *Comércio eletrônico vai faturar U$23,1 bilhões na AL até 2010*, IDG Now!, 2006.

[32] Moreira, D. *Brasil atingiu 5,75 milhões de e-consumidores no 1° semestre de 2006*, IDG Now!, 2006.

[33] *e-Commerce cresce 76% e alcança R$4,4 bilhões em 2006*. COMPUTERWORLD, IDG Now!, 2007.

[34] Pressman, R. S. *What a tangled Internet We Have*. IEEE Software, 2000.

[35] Hansen, S., Deshpande, Y., Murugusan, S., Ginige A. *Web Engineering: A New Discipline For Development of Web-based Systems*. First Workshop on Web Engineering in International Conference on Software Engineering, 1999.

[36] Gibbis, W. *Software's chronic crisis*. Scientific American, September, 1994.

[37] Kock, N. *A Comparative Study of Methods for Hypermedia Development*. Technical Report, 1999.

[38] Garzotto, F., Mainetti, L., Paolini, P. *Hypermedia design Analysis*. ACM, 1995.

[39] Isakowitz, T., Stohr, E. A., Balasubramanian, P. *RMM: A Methodology for Structured Hypermedia Design*. ACM, 1995.

[40] Garzotto, F., Paolini, P., Schwabe, D. *HDM: A model-based approach to Hypertext application design*. ACM, 1993.

[41] Schwabe, D., Rossi, G., Barbosa, S. D. J. *Systematic hypermedia application design with OOHDM*. ACM, 1996.

[42] Carneiro, L., Cowan, D., Lucena, C. *Introducing ADVcharts: A graphical specification of abstract data views*. CASCON´93, 1993.

[43] Harel, D. *Statecharts: a visual formalism for complex systems*. Science of Computer Programming, 1987.

[44] Kroll, P. *Who will benefit from the Eclipse Process Framework*. Eclipse Foundation.

[45] Balduino, R., Lyons, B. *OpenUP/Basic – A Process for Small and Agile Projects*. Eclipse Foundation.

[46] Haumer, P. *Eclipse Process Framework – Part 1: Key Concepts*. Eclipse Foundation.

[47] Haumer, P. *Eclipse Process Framework – Part 2: Authoring method content and process*. Eclipse Foundation.

[48] Agile Manifesto. Disponível em: http://www.agilemanifesto.org. Último acesso em 26/02/2007.

[49] Sommerville, I. *Engenharia de Software*, Addison Wesley, 2003.

[50] *World Wide Web Consortium*. Disponível em: http://www.w3.org, Último acesso em: 28/03/2006.