Pós-Graduação em Ciência da Computação

"XWEBPROCESS: UM PROCESSO ÁGIL PARA O DESENVOLVIMENTO DE APLICAÇÕES WEB"

Por

AMÉRICO TADEU FALCONE SAMPAIO

Dissertação de Mestrado

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

RECIFE, MARÇO/2004

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

AMÉRICO TADEU FALCONE SAMPAIO

"XWEBPROCESS: UM PROCESSO ÁGIL PARA O DESENVOLVIMENTO DE APLICAÇÕES WEB"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR(A): PROF. DR. ALEXANDRE MARCOS LINS DE VASCONCELOS

RECIFE, MARÇO/2004

AGRADECIMENTOS

Primeiramente a Deus, nosso criador e maior ponto de apoio nos momentos difíceis.

A minha amada esposa Ana Beatriz que tanto me apóia e ajuda nos momentos mais difíceis. Seu amor, carinho e dedicação foram fundamentais nesta caminhada.

Aos meus pais Pedro Dario e Rafaela pelo amor, ajuda, educação e carinho dado em toda a minha vida. Aos meus irmãos Pedro Rafael e Dariella pelo apoio e ajuda que sempre me dão.

Aos grandes amigos que fiz durante o mestrado sem ordem de preferência: Cléber (Zanche), Tarcísio (Pinto), Joel, Marco Antônio e Júnior.

Aos alunos da graduação do Centro de Informática da UFPE que participaram do experimento na disciplina Engenharia de Software. A todos os outros colegas que me auxiliaram durante todo o curso.

Ao meu orientador professor Alexandre Vasconcelos pelo excelente trabalho e pelo incentivo em buscar sempre a qualidade no trabalho. Muito obrigado!

Aos professores Paulo Borba, Augusto Sampaio, Nelson Rosa e Carlos Ferraz pelos ensinamentos e outros tipos de apoio.

Aos professores Hermano Perrelli e Jones Albuquerque pelas contribuições dadas na avaliação deste trabalho.

XWEBPROCESS: UM PROCESSO ÁGIL PARA O DESENVOLVIMENTO DE APLICAÇÕES WEB

Autor: Américo Tadeu Falcone Sampaio
Orientador: Prof. Dr. Alexandre Marcos Lins de Vasconcelos

RESUMO

O crescimento da internet e da World Wide Web provocou mudanças em muitos aspectos da nossa sociedade. A Web veio a oferecer novas maneiras de fazer negócio e muitas organizações passaram a oferecer seus serviços e migrar seus sistemas para este ambiente. As aplicações Web apresentam características importantes que devem ser tratadas adequadamente por técnicas, métodos e processos da engenharia de software. Na maioria dos casos, as restrições envolvidas nos prazos de entrega de sistemas Web são críticas por se tratar de uma questão estratégica do negócio do cliente. Neste trabalho apresentamos um processo ágil de desenvolvimento de aplicações Web, XWebProcess, que é baseado em Extreme Programming (XP) e cujo foco é a construção de sistemas Web de qualidade de forma eficiente e rápida. O processo é descrito usando a meta linguagem SPEM para facilitar seu entendimento e melhorias futuras. Um experimento foi conduzido com um grupo de quarenta estudantes de graduação para avaliar a eficiência e qualidade do processo proposto. Os resultados mostraram que XWebProcess foi tão ágil quanto XP e também que o processo é bastante adequado para o desenvolvimento Web considerando questões como análise de requisitos, projeto de navegação e interface gráfica, e testes.

Palavras-chave: Métodos ágeis, Extreme Programming, Engenharia de Software para a Web, XWebProcess, SPEM, Experimentos.

XWEBPROCESS: UM PROCESSO ÁGIL PARA O DESENVOLVIMENTO DE APLICAÇÕES WEB

Author: Américo Tadeu Falcone Sampaio
Adviser: Dr. Alexandre Marcos Lins de Vasconcelos

ABSTRACT

The growth of the Internet and the World Wide Web has contributed to significant changes in many areas of our society. The Web has provided new ways of doing business and many companies have been offering new services and migrating their systems to the Web. Furthermore, Web applications present important characteristics that must be properly addressed by software engineering methods, processes and techniques. In most cases, Web systems have to be delivered in severe time constraints due to its strategic nature for the client's business. In this work we describe an agile process for Web-based application development, XWebProcess, which is based on Extreme Programming and aims at building high quality Web applications in a time effective way. The process was described using the SPEM meta-modeling language to facilitate its understanding and further improvements. An experiment was conducted with a group of forty senior undergraduate students to assess the time effectiveness and the quality of the proposed process. The results have shown that XWebProcess is equally agile when compared to Extreme Programming, moreover, surveys conducted as part of the experiment pointed out that XWebProcess is suitable to Web development in dimensions such as requirements gathering, user interface and navigation design, and software testing.

Keywords: Agile Methods, Extreme Programming, Web Engineering, XWebProcess, SPEM, Experimentation.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

1
Introdução

Este capítulo apresenta uma visão geral desta dissertação. A Seção 1.1 apresenta a motivação deste trabalho. A Seção 1.2 demarca o escopo deste trabalho bem como as contribuições esperadas. Finalmente, a Seção 1.3 fornece uma visão dos capítulos da dissertação.

1.1 Motivação

A crescente demanda por software de qualidade vem sendo apontada como um dos maiores desafios da área de engenharia de software ao longo de vários anos [44,47]. As empresas são cada vez mais dependentes de software para a execução de seus processos de negócio, para a viabilização dos serviços prestados e para a implementação de políticas de tratamento customizado aos clientes.

Nos últimos anos, o crescimento do uso da Internet, intranets e da World Wide Web contribuiu de forma significativa para diversas mudanças na sociedade. A Web ofereceu uma nova forma de fazer negócios e a partir de então diversas empresas passaram a migrar seus sistemas e oferecer serviços pela Web.

As aplicações Web possuem algumas características especiais que demandam um tratamento adequado por parte de seus processos de desenvolvimento como, por exemplo, tratamento de requisitos não funcionais (concorrência, segurança, carga na rede), envolvimento de diversas tecnologias de desenvolvimento, interface gráfica mais rica, e testes mais complexos [5,11,49].

Apesar desta complexidade maior, na maioria dos casos a abordagem adotada para a construção de aplicações Web ainda é ad-hoc [8,9]. Nestes casos, o sucesso do

projeto depende muito das habilidades e conhecimento da equipe de desenvolvimento e acaba normalmente requerendo esforço adicional para cumprir prazos e orçamentos.

Um dos principais fatores responsáveis por essa forma desordenada no desenvolvimento Web é o tempo de entrega. Como as aplicações Web geralmente representam oportunidades de negócio estratégicas para a empresa, o tempo de desenvolvimento e entrega normalmente é curto devido à concorrência existente no mercado.

Portanto, é necessário que os processos de desenvolvimento de software existentes se adaptem para tratar de forma adequada as características especiais de aplicações Web e, além disso, que sejam eficientes do ponto de vista de tempo de entrega do sistema.

1.2 Escopo e contribuições esperadas

Neste trabalho propomos um processo ágil de desenvolvimento de software chamado XWebProcess. Este processo consiste em uma adaptação de Extreme Programming (XP) [22,28,29] para o desenvolvimento de aplicações Web. A adaptação procura considerar as fraquezas e virtudes de XP em relação à natureza Web e prover um meio eficaz de construir tal tipo de aplicação. Os passos dados para a elaboração deste trabalho são mostrados a seguir.

Primeiramente foi feita uma análise de quais características das aplicações Web precisam ser tratadas por um processo de desenvolvimento. Esta análise foi feita considerando as principais disciplinas de um processo de desenvolvimento como: requisitos, análise e projeto, implementação e testes. Esta análise apontou então o que um processo deveria conter sob um ponto de vista da Web.

Depois disso, o processo XP foi estudado dentro da ótica Web para apontar quais as necessidades de adaptação. Para se ter uma descrição melhor da estrutura do

processo foi feita a modelagem de XP usando SPEM[1] [48]. Além de melhorar o entendimento de XP, a modelagem serviu para facilitar as adaptações de XP ao desenvolvimento Web que se deram através da criação de XWebProcess. Tais adaptações consistiram na inserção e modificação de alguns elementos de processo em XP como: disciplinas, atividades, papéis e artefatos. Finalmente, um estudo experimental foi conduzido para validar XWebProcess.

A principal contribuição esperada para este trabalho é prover um processo eficaz para a construção de aplicações Web. O processo deve não só ser capaz de tratar adequadamente as características das aplicações Web, mas também deve ser ágil contribuindo para a entrega rápida do sistema. Além disso, espera-se que a iniciativa de modelar o processo usando SPEM sirva para facilitar futuras adaptações de XWebProcess para tratar questões não previstas neste trabalho.

1.3 Estrutura da Dissertação

Além deste capítulo inicial esta dissertação possui mais seis capítulos e três apêndices organizados da seguinte forma:

O **Capítulo 2** apresenta uma visão geral sobre a engenharia de software para a Web. São mostradas características gerais das aplicações Web, além de técnicas de projeto e tecnologias para a Web.

O **Capítulo 3** descreve os principais conceitos relativos a processos de software e sobretudo XP. O processo XP é detalhado e são apresentados suas práticas, princípios e papéis.

O **Capítulo 4** mostra a modelagem feita de XP utilizando SPEM. São apresentados aspectos relativos à modelagem de processos de software e também uma descrição geral dos conceitos e estereótipos de SPEM. O processo XP é modelado

[1] Software Process Engineering Metamodel (SPEM) é uma linguagem padrão da OMG para modelagem de processos de software.

segundo as perspectivas dinâmica e estrutural através de diagramas UML estereotipados.

O **Capítulo 5** descreve o processo XWebProcess. É mostrada uma contextualização de XP para o desenvolvimento Web além de como e porque XP foi usado para a concepção de XWebProcess.

O **Capítulo 6** descreve o experimento conduzido para a validação de XWebProcess. São mostrados como foram feitos a organização do experimento e também a análise e interpretação dos resultados obtidos.

O **Capítulo 7** apresenta a conclusão deste trabalho. São descritas as principais dificuldades encontradas, os trabalhos relacionados, as contribuições e também as perspectivas de trabalhos futuros.

O **Apêndice A** mostra uma descrição geral da meta-linguagem SPEM. São descritos os principais conceitos relativos à definição e estrutura de SPEM.

O **Apêndice B** apresenta a modelagem completa de XP que é descrita de forma parcial no Capítulo 4.

O **Apêndice C** mostra alguns modelos de artefatos e documentos que foram utilizados durante o experimento descrito no Capítulo 6.

2

Engenharia de software para a Web

Este capítulo apresenta o estado da arte da engenharia de software para a Web. A seções 2.1 e 2.2 mostram, respectivamente, uma visão geral sobre a Web e sobre a engenharia de software para a Web. A Seção 2.3 procura levantar considerações importantes sobre aplicações Web, e a Seção 2.4 descreve técnicas de modelagem e projeto de sistemas Web. A Seção 2.5 apresenta as principais tecnologias e plataformas de desenvolvimento de aplicações Web e a Seção 2.6 mostra alguns processos de desenvolvimento para aplicações Web. A Seção 2.7 descreve um processo de referência para desenvolvimento Web. Finalmente, a Seção 2.8 mostra as considerações finais do capítulo.

2.1 Introdução à Web

Desde sua criação nos anos 90 até os dias de hoje, a Web[2] ganha uma crescente popularidade. Ela proporciona uma nova forma de fazer negócios e de oferecer serviços que atingem um público vasto em uma área geograficamente ilimitada. Além disso, sua adoção pode trazer diversos benefícios para as organizações como reduzir custos dos processos de negócio, oferecer novos tipos de serviços, melhorar a qualidade dos produtos e facilitar a implementação de serviços customizados para clientes.

No entanto, sua adoção por parte das empresas e organizações não é uma tarefa simples. Para se entrar no mundo da Web, é necessário uma reengenharia dos processos

[2] Web ou World Wide Web (WWW) [1,2,3]: aplicação que possibilita a integração de documentos hipertexto, localizados em servidores interconectados pela internet, que podem ser visualizados por programas especiais chamados navegadores (browsers).

de negócio da empresa visando uma adequação às características dos diversos tipos de tecnologias que envolvem o ambiente Web. Antes de resolver simplesmente desenvolver uma aplicação Web para automatizar certo procedimento ou oferecer certo serviço, a empresa deve se preocupar com algumas questões, tais como:

- Como esta nova tecnologia pode beneficiar o resultado final e fazer o negócio mais rentável?
- Quais tipos de hardware, software, e serviços integrados devem ser utilizados na empresa e nos seus parceiros de maneira que permitam flexibilidade em casos de mudanças dos processos de negócio e das tecnologias?

Para responder a tais perguntas de forma eficiente e obter sucesso com a Web, existem dois tipos de preocupação que devem ser abordados na solução do problema. No nível estratégico, devem existir procedimentos e métodos eficientes de gerência na organização além de uma excelente compreensão do domínio do negócio. No nível operacional, devem existir padrões e metodologias para a produção, uso ou aquisição de sistemas Web. Além disso, é importante a utilização de uma infra-estrutura tecnológica adequada para dar suporte aos Websites produzidos ou utilizados pela empresa.

2.1.1 Áreas de aplicação da Web

Quando a Web foi criada, seu principal intuito era o de compartilhar informações ao redor do mundo servindo como uma fonte de pesquisa. Com o passar do tempo, software e serviços para facilitar o acesso a Web foram sendo desenvolvidos e oferecidos, popularizando a Web. Empresas, órgãos governamentais e pessoas de dentro de suas casas passaram a se interessar e utilizar a Web como ferramenta de pesquisa, como forma de vender e oferecer serviços e como uma forma de reorganizar processos de negócio.

Existem diversos usos da Web que devem ser mencionados devido à sua importância e utilidade tais como:

- Na área de educação ela pode ser usada para aprendizado à distância (e-learning). Algumas universidades como o conceituado Massachussets Institute of Technology (MIT) já oferecem seus cursos pela Web. Websites oferecem tutoriais, bibliotecas digitais, engenhos de busca e outros serviços;
- Na área de negócios ela pode oferecer oportunidades de venda de produtos e serviços (comércio eletrônico) como as famosas lojas virtuais (Amazon e Americanas) onde os clientes podem comprar desde livros até eletro-eletrônicos. São também bastante populares os sites de leilão virtual (Mercadolivre, eBay e Arremate) onde os usuários podem fazer lances e ofertas em produtos de diversos tipos;
- Na área de entretenimento existem jogos em rede onde diversas pessoas podem participar simultaneamente, simulando situações como partidas de futebol, guerras e corridas de automóvel;
- Na área de divulgação de notícias e informações existem os jornais e revistas, os grupos de discussão e notícias e fóruns virtuais. A maioria dos grandes jornais (New York Times, Folha de São Paulo) canais de televisão (NBC, CNN, Globo) e revistas (Scientific American, Veja) do mundo já disponibilizam suas informações em sites.
- Na área de prestação de serviços a é muito comum o uso da Web em sistemas de transações bancárias (home banking) onde os clientes de diversos bancos (Bank of Boston, Banco do Brasil, Caixa Econômica) podem fazer depósitos, transferências, verificar extratos e pagar contas.

Isso traz diversos benefícios para as empresas, governos e pessoas. As empresas podem reduzir custos nas operações, atingir um público muito maior para vender seus serviços e ter um novo veículo de divulgação da sua marca. O setor público pode oferecer serviços pela Web, que antes eram relativamente custosos, como, por exemplo, declaração de imposto de renda, emissão de documentos, inscrição em concursos públicos e pagamento de taxas, economizando tempo e dinheiro, e facilitando a vida dos

usuários. As pessoas ganham um novo veículo de pesquisa de informações, uma nova "loja" global que oferece produtos de diversos tipos e de vários locais do mundo, um novo veículo para aprendizado e estudo, e também uma nova forma, muito simples, de utilizar serviços para resolver facilmente seus problemas.

2.2 Visão geral da Engenharia de Software para a Web

O crescimento do uso da internet, intranets, extranets e da World Wide Web tem provocado um impacto significativo em diversos setores da sociedade. À medida que cresce a sofisticação e a quantidade de aplicações Web, surge a preocupação com a maneira com que essas aplicações são construídas, com a sua qualidade e integridade no longo prazo.

A complexidade da construção dos sistemas Web está relacionada a alguns fatores como: uso de diversas tecnologias necessárias para criar interfaces gráficas complexas que podem conter sons, figuras, texto e imagens; suporte a mecanismos de computação distribuída como processamento remoto, concorrência e balanceamento da carga e do acesso as bases de dados; e manutenção dos componentes que fazem parte do sistema (classes de negócio, páginas, figuras, imagens e bases de dados).

Em muitos casos, as abordagens utilizadas no desenvolvimento de software para a Web foram e ainda são ad hoc [9]. Nesses casos, o sucesso do projeto depende muito da qualidade e desempenho das pessoas envolvidas e acaba quase sempre exigindo muito esforço e trabalho extra das pessoas para cumprir os prazos estabelecidos. Observa-se então que o desenvolvimento de software para a Web carece de uma metodologia mais rigorosa e sistemática [5,83].

Existe uma preocupação crescente de que poderão ocorrer sérios problemas relativos a desenvolvimento, distribuição e manutenção de sistemas Web [6,7,8,83]. Um grande número de aplicações Web que estão sendo feitas de forma inadequada agora possuem uma grande probabilidade de falhar e causar danos irreparáveis no

futuro. Além disso, como sistemas Web muitas vezes implementam negócios estratégicos da organização e podem integrar vários sistemas, uma falha em um sistema deste porte pode causar muitos danos e prejuízos. Caso isso aconteça, a confiança na Web pode ser abalada de forma a causar uma "crise na Web" [5,8,9].

Com o objetivo de evitar uma crise na Web e de atingir sucesso no desenvolvimento de aplicações complexas para a Web, surge uma necessidade cada vez maior de abordagens disciplinadas para a engenharia dessas aplicações.

Métodos, ferramentas e processos para desenvolver, distribuir, avaliar e gerenciar sistemas Web devem ser criados ou adaptados para a realidade da Web. O importante é que tais abordagens e técnicas devem levar em consideração as características especiais da Web, os ambientes operacionais, os cenários e a multiplicidade de perfis de usuário que impõem desafios adicionais no desenvolvimento de aplicações Web.

A engenharia de software para a Web (Web Engineering) se preocupa com o estabelecimento e uso de princípios confiáveis e científicos de engenharia e gerência, além de abordagens disciplinadas e sistemáticas para o sucesso no desenvolvimento, distribuição e manutenção de aplicações Web de alta qualidade. Isto deve englobar diversas atividades, começando desde a concepção e desenvolvimento e indo até implementação, avaliação de performance e manutenção contínua.

Os principais tópicos de interesse da engenharia de software para a Web (Web Engineering), conforme descrito em [5], incluem:

- Especificação e análise de requisitos;
- Metodologias, técnicas e processos de desenvolvimento de sistemas para a Web;
- Integração com sistemas legados;
- Migração de sistemas legados para o ambiente Web;
- Gerência de informação;
- 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;
- Impacto global e social da Web.

Diante de tantas questões, o presente trabalho se concentrará nas metodologias e processos de desenvolvimento para a Web. Isso será tratado no processo XWebProcess e será visto ao longo dos próximos capítulos. A próxima seção aborda algumas questões importantes sobre sistemas para a Web que devem ser consideradas.

2.3 Considerações importantes sobre sistemas Web

A engenharia de software se preocupa em estabelecer princípios, técnicas, métodos e processos para que o objeto de seu estudo, no caso software, seja concebido de maneira apropriada e eficiente, respeitando-se suas características intrínsecas e as restrições que normalmente cercam o processo de construção, tais como prazos, orçamento e qualidade do produto.

Dentro desta perspectiva, a engenharia de software para a Web deve procurar avaliar quais as características de sistemas Web que precisam de tratamentos específicos, de modo que as técnicas, métodos e processos existentes na engenharia de software possam se adaptar para tratá-las de forma adequada. Algumas destas características descritas em [6,8,9,12,15] são:

- As questões referentes a requisitos não funcionais como concorrência, distribuição, balanceamento de carga, escalabilidade, segurança e persistência são muito importantes em aplicações Web;
- Diferentes tipos de clientes podem acessar os sistemas via browsers, protocolos e máquinas distintas (celulares, PCs, palms, etc...);
- Aplicações Web rodam em ambientes heterogêneos com muitos componentes de hardware e software distintos;
- As questões relativas à navegação[3], apresentação e interfaces gráficas são complexas em sistemas Web (podem possuir gráficos, sons, imagens, vídeos, etc.) e demandam um processo de criação adequado;
- Os sistemas Web podem lidar com diversos tipos de cliente com perfis distintos e implementar customizações específicas dependendo do perfil;
- O conteúdo de alguns sistemas Web pode mudar bastante como, por exemplo, em sites de revistas e jornais onde o conteúdo pode ser alterado diversas vezes por dia;
- A arquitetura de um sistema Web deve ser flexível e bem estruturada, pois os sistemas Web estão constantemente sendo alterados, integrados e incorporando novas tecnologias.

Ao observar estas características pode-se notar que grande parte delas são intrínsecas à natureza das aplicações Web e, portanto elas representam novos desafios que devem ser tratados pela engenharia de software. Na próxima seção serão mostradas algumas técnicas de projeto de sistemas Web que surgiram recentemente e já procuram lidar com algumas destas questões.

[3] É o processo pelo qual o usuário da aplicação Web "navega" pelo conteúdo do site através das suas páginas e links.

2.4 Técnicas de modelagem e projeto de sistemas Web

Nesta seção serão apresentadas algumas técnicas de modelagem e projeto de sistemas Web. Estas técnicas procuram tratar as questões relativas a interfaces gráficas e navegação. Na Seção 2.4.1 serão apresentadas técnicas de modelagem de sistemas Web baseadas em UML [10,36,37].

Na Seção 2.4.2 serão mostradas algumas técnicas de projeto de sistema Web que não são baseadas em UML e que levam em consideração questões acerca da navegação e interfaces gráficas de sistemas Web. Na Seção 2.4.3 são descritas algumas técnicas mais informais de projetar navegação e interfaces gráficas de sistemas Web. Além disso, na Seção 2.4.4 são feitos breves comentários sobre alguns conceitos muito usados em projetos de sistemas Web como arquitetura em camadas, padrões de projeto e padrões arquiteturais.

2.4.1 Técnicas de projeto de sistemas Web baseadas na UML

A UML *(Unified Modeling Language*) [10,36,37] é uma linguagem padrão para modelagem de sistemas, principalmente os orientados a objetos. Ao longo dos últimos anos, surgiram diversas técnicas baseadas na UML para tratar a questão da modelagem de sistemas hipermídia (Web). Essas técnicas se baseiam em alguns conceitos comuns, que são tratados de formas diferentes por cada técnica [71]. Estes conceitos são mencionados abaixo:

- **Modelos do domínio da aplicação:** Os modelos do domínio procuram representar as informações que fazem parte das regras de negócio do sistema;
- **Modelos navegacionais:** Estes modelos procuram representar as diversas maneiras que o usuário pode navegar pela rede de informações

de um sistema Web. Grupos de informações similares são agrupados em um mesmo contexto e interconectados por “links”;

- **Modelos de apresentação:** Estes modelos procuram tratar a aparência das informações apresentadas para os usuários.

Algumas das principais técnicas que se baseiam nesses conceitos são: WAE [12], UHML [13], W2000 [14] e WebML [72]. Estas técnicas definem extensões na UML para tratar os diversos tipos de modelos descritos anteriormente e usam mecanismos da UML como diagramas de classes, de estados, casos de uso e outros, para modelar os sistemas Web.

Um modelo produzido em [11] pode dar uma idéia de como estas técnicas podem ser usadas. A Figura 2.1 apresenta um modelo elaborado com a técnica WAE que representa, através de um diagrama de classes estereotipado, o relacionamento entre as classes de interface gráfica do sistema. O diagrama define que a página cliente “InformacaoComprador” agrega o formulário “DadosComprador”, o qual possui diversos campos de entrada e seleção e cujos dados serão submetidos a uma página servidora para processamento.

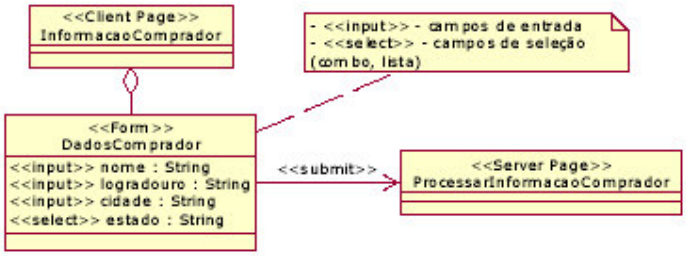


Figura 2.1 – Exemplo de modelagem usando estereótipos de WAE

2.4.2 Técnicas de projeto de sistemas Web não baseadas em UML

Além dos métodos apresentados na seção anterior, outros métodos que não se baseiam em UML também surgiram para resolver os problemas de projetar e modelar aplicações Web com ênfase em questões de navegação e apresentação. Alguns destes trabalhos podem ser vistos em [17,18,38,73,74] e, na sua maioria, são métodos de

análise e projeto orientado a objetos. Existem diversas técnicas para modelagem de sistemas Web não baseadas em UML.

Algumas das principais técnicas dentro desta categoria são: HDM [73], RMM [74], OOHDM [18] e OOWS [38]. Não convém descrever neste trabalho cada uma destas técnicas, porém é importante dar uma visão geral sobre as mesmas.

- HDM e RMM estão entre as técnicas mais antigas para o projeto de sistemas hipermídia e se baseiam na notação E-R (entidade e relacionamento) para a construção dos modelos de domínio e navegacionais. Foram as primeiras técnicas a criar a noção de entidades navegacionais e outros elementos da modelagem hipermídia como elos (links) e nós (nodes);
- OOHDM é uma técnica OO que se baseou nas duas anteriores. Uma aplicação hipermídia (Web) é construída em um processo de cinco passos em um estilo iterativo e incremental. As cinco atividades de OOHDM são: captura de requisitos, modelagem de domínio ou conceitual, projeto navegacional, projeto da interface abstrata e implementação;
- OOWS é uma extensão Web de um método de projeto orientado a objeto chamado OO-Method. As questões de modelagem conceitual, navegacional e apresentação são tratadas de forma semelhante às técnicas anteriores, através de extensões criadas para o OO-Method. Além disso, o método possui o suporte de ferramentas que auxiliam na criação dos modelos e na geração de parte do código aplicação Web.

2.4.3 Técnicas “informais” de projeto de sistemas Web

As duas seções anteriores citaram métodos de projeto de sistemas Web que, em sua maioria, se baseiam em outros métodos orientados a objetos que já foram pesquisados e que possuem uma certa credibilidade científica.

Com o advento dos processos ágeis de desenvolvimento como Extreme Programming [28,29] e seu sucesso obtido em vários projetos, diversos pesquisadores das áreas industrial e acadêmica passaram a questionar a eficiência de métodos mais formais[4] em projetos de desenvolvimento de software considerando principalmente que o custo de usar tais métodos não traz benefícios suficientes para justificar seu uso. Ou seja, eles argumentam que o esforço usado para produzir os modelos poderia ser redirecionado para produzir modelos menos complexos, que contenham apenas as informações mais relevantes, e para produzir código, que é o enfoque principal. Diversos conceitos então começaram a surgir como "Agile Requirements", "Agile Modeling" que procuravam estabelecer técnicas mais "informais" para diversas disciplinas da engenharia de software (Requisitos, Análise e Projeto, etc.) com o intuito de construir software de qualidade com custos menores e em menos tempo.

Esta seção procura apresentar uma técnica mais "informal" que poderia servir para o projeto de navegação e apresentação de interfaces gráficas de sistemas Web. A técnica foi descrita por Scott Ambler em [34] e se baseia simplesmente em desenhar em um quadro branco ou numa folha de papel um protótipo das páginas Web, representando a navegação entre elas através de flechas que indicam o sentido da navegação.

Ambler ainda sugere que se o artefato tivesse que ser guardado como parte da documentação do sistema, poder-se-ia bater uma foto digital do desenho no quadro branco ou escanear o rascunho feito em papel. O projeto da navegação e apresentação do sistema Web poderia ser criado em questão de minutos e contar com a sugestão de várias pessoas da equipe de desenvolvimento, numa espécie de sessão de "brainstorming". Um exemplo de um projeto de navegação e interface gerado dessa forma pode ser visto na Figura 2.2.

[4] Formal aqui no sentido de estruturado, científico e não no sentido de especificação formal.

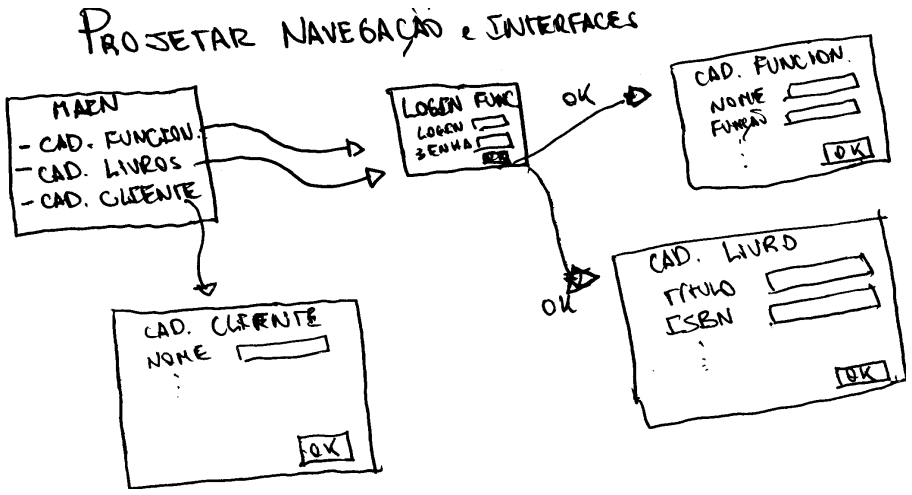


Figura 2.2 - Projeto da Navegação e Interfaces de sistema Web (documento escaneado)

Outra técnica que pode ser utilizada principalmente para esclarecer as funcionalidades, estruturar as páginas Web e ajudar na elicitação de requisitos de sistemas Web é a técnica de prototipagem "White Site Prototyping", sugerida por Henderson-Sellers em [39]. Esta técnica consiste basicamente em construir um protótipo do "site" em branco (sem figuras, cor, etc.) para esclarecer requisitos junto ao cliente, minimizar as dúvidas sobre o domínio do problema e, além disso, dar uma idéia inicial sobre a estruturação dos elementos da aplicação Web.

As técnicas "informais" apresentadas nesta seção são importantes para este trabalho, pois foram usadas no estudo de caso do processo XWebProcess para fazer o projeto da navegação e interface gráfica do sistema que foi implementado.

2.4.4 Arquiteturas e padrões de projeto para a Web

A arquitetura de um software corresponde a uma visão que representa sua estrutura ou esqueleto. Um modelo comum e bastante adotado em sistemas Web conforme visto em [40,41,42] é o de arquitetura em camadas (layers). A idéia deste modelo é o de separar os aspectos relativos à apresentação (interface com o usuário), comunicação (distribuição), negócio e dados. Tal estruturação do sistema permite que seja minimizado o entrelaçamento de códigos com diferentes propósitos, como por exemplo código de negócio com código de comunicação, e juntamente com o uso de

padrões de projeto, permite alcançar melhores níveis de reuso além de facilitar a manutenção do sistema. Um exemplo de arquitetura em camadas pode ser visto na Figura 2.3.

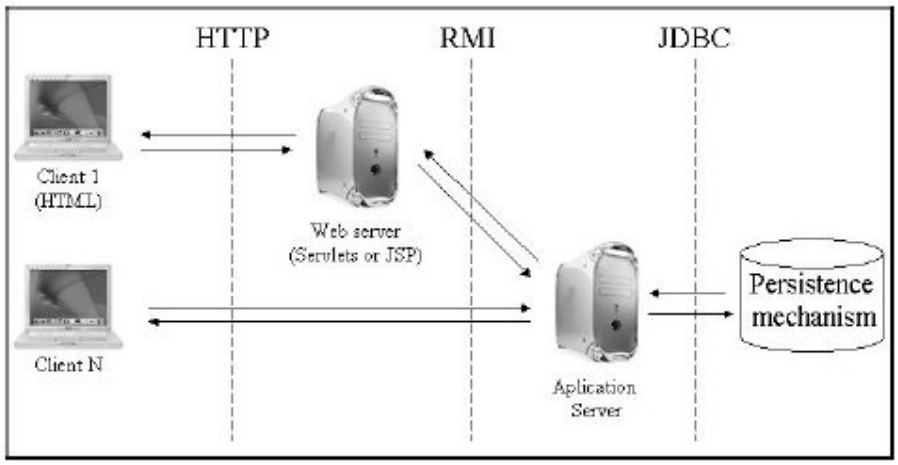


Figura 2.3 – Exemplo de arquitetura em camadas

Aqui os clientes que usam o sistema Web através de seus navegadores (browsers), ao acessar um Web site fazem solicitações via HTTP aos servidores Web ou diretamente ao servidor de aplicação onde está o código de negócio, e este último se comunica com a camada de dados. Um exemplo de padrão de projeto que pode ser usado para implementar arquiteturas em camadas é descrito em [43]. Uma explicação mais detalhada sobre as tecnologias envolvidas neste processo será feita na próxima seção.

2.5 Principais tecnologias e plataformas para a Web

Esta seção mostra as principais tecnologias utilizadas para a implementação de sistemas Web (Seção 2.5.1) bem como as principais plataformas existentes hoje em dia para a construção de aplicações Web de grande porte (Seção 2.5.2).

2.5.1 Principais tecnologias

A rápida expansão que a Web alcançou ao longo dos últimos anos se deve em grande parte ao surgimento de várias tecnologias para a implementação de aplicações Web. Esse fenômeno se deu devido ao interesse de grandes empresas como Microsoft, IBM, Sun e outras que vislumbraram a Web como estratégica para os seus negócios e com isso passaram a investir na criação de tecnologias para dar suporte a construção desse tipo de aplicação.

Devido ao grande número de tecnologias existentes para o desenvolvimento Web convém apresentar uma visão geral destas. Para isso, é apresentada a Tabela 2.1 onde as diversas tecnologias são agrupadas de acordo com as camadas lógicas apresentadas na seção anterior.

Tabela 2.1 - Tecnologias de implementação Web

CAMADA	TECNOLOGIAS	FUNÇÃO
Camada de apresentação	HTML, JavaScript, VBScript, Figuras, Áudio, Vídeo, JSP, Servlets, ASP, etc.	Estas tecnologias servem para construir as interfaces gráficas das aplicações Web. Podem englobar processamento do lado do cliente como em linguagens de Script (Ex: JavaScript, VBScript) ou processamento do lado do servidor para geração de conteúdo dinâmico (Ex: ASP, JSP, Servlets).
Camada de comunicação	Middlewares (CORBA, RMI, EJB, Web Services, DCOM), protocolos de rede (HTTP, TCP/IP), etc.	São tecnologias que facilitam a comunicação dos componentes de software (aplicações, objetos) distribuídos pela rede, mascarando certos detalhes de mais baixo nível e fornecendo APIs de mais alto nível para o programador.

CAMADA	TECNOLOGIAS	FUNÇÃO
Camada de negócio	Java, Visual Basic, C#, etc.	A função dessas tecnologias é implementar os objetos de negócio da aplicação. A comunicação da camada de apresentação com essa camada pode ser viabilizada por meio da camada de comunicação e de tecnologias de processamento do lado do servidor vistas anteriormente.
Camada de dados	APIs (JDBC, OLEDB, ADO.NET) para comunicação com SGBS (Oracle, SQLServer, etc.)	Essas tecnologias servem para oferecer APIs que permitam fazer a comunicação entre os objetos de negócio e os mecanismos de persistência de forma fácil para o programador.

2.5.2 Principais plataformas

Com o surgimento das tecnologias mostradas na seção anterior, empresas como Microsoft e Sun Microsystems tiveram a idéia de criar o que tem se chamado de plataformas de desenvolvimento para construção de aplicações Web. Essas plataformas se constituem de um conjunto de tecnologias, cada uma com uma função bem definida, que servem para a construção de aplicações Web de grande porte como sites de comércio eletrônico "business to business" e "business to consumer" (B2B e B2C). Essas plataformas procuram oferecer tecnologias e mecanismos que tratam de questões importantes em grandes aplicações Web como segurança, escalabilidade, comunicação remota, gerenciamento de transações e interoperabilidade.

Diante disso, duas grandes plataformas surgiram e passam hoje a disputar esse segmento de mercado. São elas a plataforma J2EE (Java 2 Platform, Enterprise Edition) [25] desenvolvida pela Sun Microsystems e a plataforma .NET [26] da Microsoft. Estas duas plataformas apresentam algumas similaridades e diferenças e utilizam algumas das

tecnologias apresentadas na seção anterior. Não é propósito deste trabalho detalhar como estas plataformas diferem ou mesmo em que situação uma ou outra é mais adequada. Uma excelente discussão sobre isto pode ser encontrada em [27,45].

A Figura 2.4 e a Figura 2.5 apresentam uma visão arquitetural das duas plataformas e permitem mostrar as principais tecnologias usadas por cada uma delas. É importante observar que tais tecnologias desempenham papéis similares [45] no desenvolvimento de aplicações Web.

As arquiteturas mostradas respeitam a mesma arquitetura em camadas apresentada na Figura 2.3 e diferem quanto ao uso de tecnologias específicas em cada plataforma. Por exemplo, enquanto J2EE utiliza JSP e Servlets para fazer o processamento dinâmico do lado do servidor a plataforma .NET utiliza ASP.NET para o mesmo propósito.

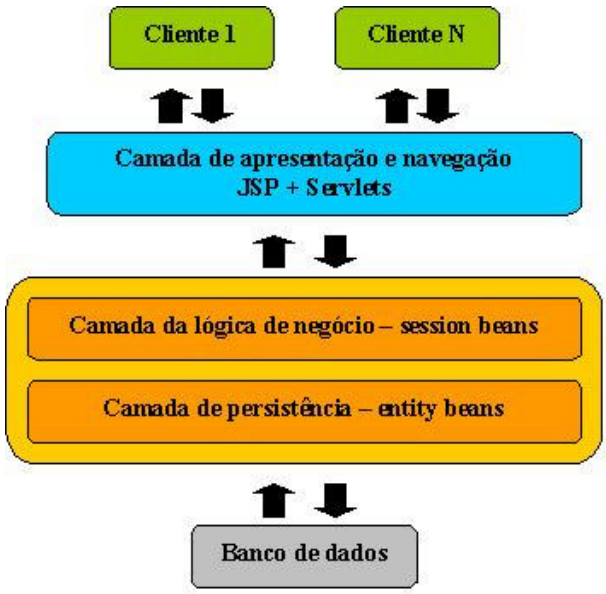


Figura 2.4 - Arquitetura de aplicação na plataforma J2EE

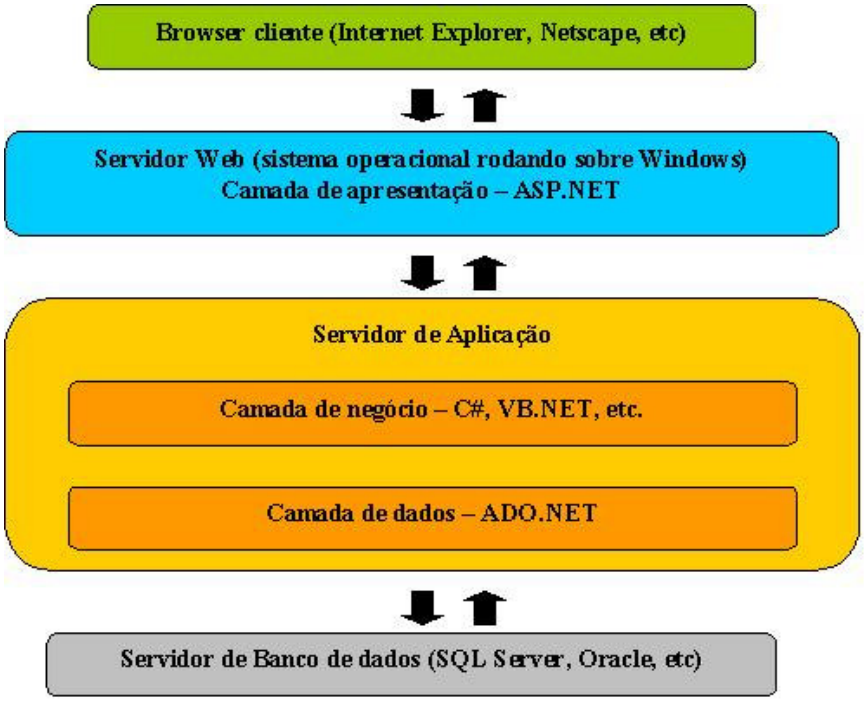


Figura 2.5 - Arquitetura de aplicação na plataforma .NET

2.6 Processos de desenvolvimento de aplicações Web

A importância dos processos de desenvolvimento de aplicações Web já foi mencionada em seções anteriores, entretanto é importante que seja delimitado o que esses processos devem abranger para que seja possível a construção de sistemas Web de alta qualidade.

Com a popularização da Web no final dos anos 90 as empresas passaram a usar a Web para fazer negócios e oferecer serviços. A demanda pela construção de aplicações Web de forma rápida, visando ganho de vantagens competitivas, não foi apoiado por uma devida adequação nos processos de software em grande parte dessas empresas acarretando alguns problemas.

Diversos problemas se devem ao fato de que as novas características das aplicações Web como interfaces gráficas mais complexas, equipes de desenvolvimento

mais multidisciplinares, restrições no ambiente de operação do Web Site (escalabilidade, carga na rede, concorrência) e requisitos mal definidos pelos clientes [4,8,9,39,45] impõem novos desafios a serem analisados e tratados pelas atuais práticas e processos existentes. Um resumo desses problemas é apresentado na Tabela 2.2 abaixo.

Tabela 2.2 - Problemas relativos ao desenvolvimento de aplicações Web

PROBLEMA	DESCRIÇÃO
Interfaces gráficas mais complexas	As tecnologias para construção de interfaces gráficas de sistemas Web envolvem diversos tipos de mídias (áudio, vídeo, gráficos, animações, etc...).
Usabilidade	A navegação dentro da aplicação Web deve ser simplificada e de fácil uso por parte dos usuários, caso contrário eles optam por Web sites de concorrentes.
Restrições do ambiente operacional	O ambiente onde o Web site está instalado deve ser capaz de atender à demanda de acesso dos seus usuários, abrangendo questões como: escalabilidade, concorrência, balanceamento de carga, segurança e outros.
Riscos para o negócio	Os sistemas Web têm uma audiência a nível mundial, de forma que um Web site mal construído pode "manchar" o nome da organização e fazer com que ela tenha prejuízos e perca mercado para concorrentes.
Equipes mais multidisciplinares	A construção de aplicações Web envolve novos papéis como, por exemplo, artistas gráficos (designers) e especialistas em Marketing que têm uma formação diferente dos desenvolvedores.
Manutenção contínua	O conteúdo de um Web site pode mudar constantemente como, por exemplo, em sites de jornais, revistas e portais, portanto precisa de mecanismos eficientes para incorporar essas mudanças.

Alguns trabalhos [4,11,39,46] desenvolvidos por diferentes pesquisadores já começaram a tratar estas questões. Estes trabalhos são brevemente discutidos aqui nas

seções 2.6.1, 2.6.2 e 2.6.3. Na Seção 2.7 é mostrado um processo de referência para o desenvolvimento de aplicações Web.

2.6.1 Agile Web Engineering (AWE) Process

O trabalho de McDonald [4,49] consiste da elaboração de um processo ágil chamado AWE (Agile Web Engineering Process) que foi feito com base em uma pesquisa realizada junto a empresas de desenvolvimento de aplicações Web do Reino Unido. Esta pesquisa procurou identificar quais eram as principais características das aplicações Web e que atividades eram necessárias a um processo de desenvolvimento Web.

Alguns dos problemas apontados anteriormente na Tabela 2.2 são mencionados nessa pesquisa onde também são apontadas algumas características comuns a muitos projetos Web que são: equipes pequenas e multidisciplinares, ciclos curtos de desenvolvimento e importância estratégica do projeto.

Com base nos problemas descobertos, McDonald definiu os principais papéis (roles), atividades e disciplinas necessários para o processo AWE. Um exemplo de papel muito importante para projetos Web, citado em [4], é o papel de “designer” criativo (Creative designer), que é responsável por criar o projeto das interfaces gráficas da aplicação Web.

O processo AWE não se baseia em nenhum processo ágil e basicamente considera o que deve ser feito em cada disciplina de desenvolvimento (requisitos, análise e projeto, implementação e testes). O processo é todo descrito de forma textual e não existe uma definição clara de como funciona sua estrutura e dinâmica, ou seja, não se tem uma idéia clara de quais papéis são responsáveis por quais atividades e nem quando uma determinada atividade começa ou termina. AWE consiste basicamente de um conjunto de disciplinas e papéis com suas respectivas descrições e que está ainda sob processo de validação através de estudos de caso.

2.6.2 Extensões do RUP para a engenharia de software para a Web

No trabalho de Souza [11] o processo Rational Unified Process (RUP) foi estendido no seu fluxo de análise e projeto para englobar um novo fluxo chamado de "Projetar Camada de Apresentação". Esse fluxo foi adaptado para utilizar técnicas de projeto de aplicações Web baseadas principalmente na técnica de Connalen [12]. Além disso, novos subfluxos como, por exemplo, "Projetar Navegação" surgiram. Nota-se que este trabalho é um pouco diferente do anterior, pois se preocupa apenas com a disciplina de análise e projeto e não com o processo como um todo. A definição das extensões fica clara tanto do ponto de vista estrutural quanto dinâmico. Um estudo de caso foi realizado para validar o processo.

O trabalho realizado em [46] é fruto da integração do RUP com o processo de "design" criativo da empresa Context, fornecedora de aplicações e serviços Web. O trabalho se preocupa principalmente com as questões de navegação e interfaces gráficas e sugere a criação de mapas navegacionais usando-se técnicas semelhantes às apresentadas na Seção 2.4. Além da criação de mapas navegacionais, outras atividades são sugeridas, como criação de um protótipo Web junto ao cliente para que ele decida sobre questões como aparência, requisitos, etc.

2.6.3 Extensão OPEN Web

O trabalho de Henderson-Sellers e Lowe [39,50] também identificou características de aplicações e projetos Web, fazendo em seguida modificações no processo OPEN. Surgiram novas atividades, tarefas, técnicas e papéis na extensão feita no processo original.

Esse trabalho é bem abrangente e aborda em que partes do processo, diversas questões importantes como gerência de conteúdo, personalização e técnicas de prototipagem para Web sites devem ser tratadas.

Neste processo existe uma definição mais clara da estrutura do processo, ou seja, de quais elementos (artefatos, disciplinas, atividades) foram modificados e aonde isso se insere no processo OPEN original. No entanto, a dinâmica do processo também é de difícil compreensão para equipes que queiram aplicá-lo em um projeto real. O processo também é descrito de forma textual e também está sendo validado através de estudos de caso. A Tabela 2.3 apresenta um quadro comparativo dos processos discutidos na Seção 2.6.

Tabela 2.3 - Quadro comparativo de processos para a Web

Característica	**PROCESSO**		
	AWE	**OPEN-Web**	**Extensões RUP**
Descrição do processo	Textual	Textual	Textual e Modelada
Escopo Abordado	Disciplinas de desenvolvimento	Disciplinas de desenvolvimento	Disciplina de análise e projeto
Facilidade de entendimento	Difícil entender estrutura e dinâmica	Fácil de entender a estrutura e difícil a dinâmica	Fácil de entender a estrutura e a dinâmica
Forma de validação	Estudo de caso ainda sendo conduzido	Estudo de caso ainda sendo conduzido	Estudo de caso já conduzido

É importante mencionar que o processo XWebProcess é descrito de forma textual e modelada usando SPEM (Capítulo 5) e seu escopo aborda as disciplinas de desenvolvimento. A modelagem facilita a compreensão tanto da sua estrutura, quanto da sua dinâmica, conforme descrito no Capítulo 6 que mostra o experimento realizado com este processo.

2.7 Um processo de referência para o desenvolvimento de aplicações Web

As pesquisas realizadas em todos os trabalhos apresentados nas seções anteriores possuem objetivos semelhantes ao desta dissertação que é o de investigar o que um processo de desenvolvimento de software para a Web deve abranger. Pode-se notar que todos os trabalhos chegaram a conclusões semelhantes sobre as principais características das aplicações Web e sobre os principais problemas relativos ao seu desenvolvimento.

Esta seção discute o que um processo de desenvolvimento de software para a Web deve conter para resolver problemas discutidos anteriormente. Conforme pode ser visto em [44,47], um processo de software se compõe de alguns elementos básicos que são: atividades, papéis, agentes (stakeholders) e artefatos. A idéia deste trabalho é investigar como um processo de software, no caso XP, pode ter seus elementos adaptados. A adaptação[5] propriamente dita só será apresentada no Capítulo 5 e terá como base o levantamento feito ao longo deste e do próximo capítulo.

Para organizar de uma forma eficiente e didática o processo, chamado de processo de referência, será usado o conceito de disciplina descrito em [48]. Uma disciplina é um pacote (componente) de processo, composto de elementos de processo (papéis, artefatos, atividades, etc...) segundo um ponto de vista como, por exemplo: análise e projeto, testes, gerência de configuração, entre outros.

A Tabela 2.4 descreve o que um processo para desenvolvimento Web deve conter em suas disciplinas. Essa tabela irá servir de base para a adaptação de XP elaborada no Capítulo 5. É importante mencionar que o processo de referência apresentado não é um processo de software completo e procura englobar apenas as disciplinas relativas ao desenvolvimento do produto e algumas disciplinas auxiliares ou de suporte.

---

[5] Não será feita uma modificação em XP, mas sim sua utilização com adaptações para a elaboração de um novo processo. Isso será melhor explicado nos Capítulos 3 e 5.

Tabela 2.4 - Disciplinas de um processo de referência ao desenvolvimento Web

DISCIPLINA	DESCRIÇÃO
Análise de domínio	Nesta disciplina devem ser feitas explorações sobre o mercado analisando produtos concorrentes, perfis de clientes, viabilidade do projeto e principais riscos. Devem participar gerentes, especialistas de domínio, clientes e gerentes técnicos.
Especificação de requisitos	Nesta disciplina devem ser feitas atividades para levantar e esclarecer os requisitos do sistema. Técnicas como entrevistas, questionários e um protótipo inicial do site podem ser utilizadas para esclarecer dúvidas com os clientes.
Análise e projeto	Nesta disciplina devem ser realizadas atividades para criar a modelagem e documentação através de técnicas de análise e projeto. Além disso, deve ser elaborada uma arquitetura do sistema que considere os problemas do ambiente mencionados na Tabela 2.2 que estarão ligados ao desempenho e segurança do sistema Web. Um projeto da navegação e da interface gráfica (páginas Web) também deve ser feito e validado junto ao cliente levando em consideração a usabilidade do sistema.
Implementação	Nesta disciplina são construídos os módulos de implementação do sistema (código em linguagem de programação, páginas HTML, Figuras, mídias, etc...). Esses diversos módulos podem ser construídos em paralelo por engenheiros de software (código) e artistas gráficos (figuras, páginas do site, animações, etc.).
Testes	Nesta disciplina é feita a validação do que foi construído durante a implementação. Testes de unidade, de integração, de sistema e de aceitação devem ser realizados. Além disso, em sistemas Web também é muito importante rodar testes que simulem situações de tráfego intenso na rede, onde muitos usuários podem estar acessando o site simultaneamente. É muito importante verificar o desempenho do sistema Web e saber do que ele é capaz antes de colocá-lo em operação. Existem ferramentas disponíveis no mercado capazes de automatizar algumas destas tarefas.
Distribuição "Deployment"	A distribuição dos arquivos que compõem a aplicação Web (HTML, Scripts, ASPs, JSPs, Figuras, etc.) é uma tarefa que requer bastante organização. Estes componentes podem

DISCIPLINA	DESCRIÇÃO
	estar distribuídos em máquinas distintas como em servidores Web, de aplicação, e de banco de dados. A desorganização nesta distribuição pode acarretar dificuldades para descobrir erros no funcionamento do sistema, além de dificultar a correção dos mesmos e a atualização das versões do sistema.
Manutenção e evolução	A manutenção e evolução de sistemas Web também tem suas peculiaridades. Os sistemas Web, como visto na Tabela 2.2, podem mudar constantemente seu conteúdo. Ferramentas devem ser utilizadas para dar suporte e automatizar tarefas de manutenção de sistemas Web como, por exemplo, ferramentas de gerenciamento de conteúdo.

Portanto, essa tabela constitui-se em um elemento importante para o restante deste trabalho e servirá como base conceitual para as extensões e adaptações que serão feitas no processo Extreme Programming (XP). A justificativa pela escolha de XP como processo base para o desenvolvimento de aplicações Web será dada no próximo capítulo.

2.8 Considerações finais

Neste capítulo foram apresentados conceitos relativos ao estado da arte da engenharia de software para a Web, os quais serão importantes para o restante do trabalho. Foram discutidas técnicas de projeto de sistemas Web que procuram tratar algumas características importantes destes sistemas como navegação e suas interfaces gráficas. Foram levantadas as principais questões e características relativas ao desenvolvimento Web que precisam ser tratadas em um processo de desenvolvimento.

Além disso, foram apresentados alguns trabalhos relativos à adaptação de processos de desenvolvimento de software para a Web cujas idéias e problemas levantados serviram para a elaboração de um modelo de referência de processo de desenvolvimento para Web. O próximo capítulo irá procurar discutir sobre o conceito de processos de software e, sobretudo sobre Extreme Programming (XP).

3
Extreme Programming (XP)

Este capítulo apresenta o processo ágil de desenvolvimento de software Extreme Programming (XP). A Seção 3.1 apresenta uma visão geral do capítulo. A Seção 3.2 apresenta uma introdução geral sobre processos de software. A Seção 3.3 explica alguns conceitos básicos de XP. A Seção 3.4 procura mostrar uma visão geral da estrutura e dinâmica de XP. A Seção 3.5 descreve quando não se deve usar XP. Finalmente a Seção 3.6 apresenta as considerações finais do capítulo.

3.1 Visão geral

Um processo de software consiste de um conjunto de atividades realizadas para a sua construção e manutenção. Ao longo dos últimos anos, a crescente demanda por software de qualidade, aliada à pressão em relação aos prazos de entrega fizeram com que a eficiência de muitas técnicas e processos tradicionalmente utilizados passasse a ser questionada.

Diante deste contexto, os processos ágeis como Extreme Programming [28, 29], passaram a atrair o interesse dos meios acadêmico e industrial. Devido à simplicidade e agilidade adotadas na solução de problemas através do uso de eficientes práticas de programação, as equipes que usam XP vêm obtendo sucesso em seus projetos o que vem contribuindo para o aumento da sua popularidade.

Este capítulo procura apresentar o processo Extreme Programming. São apresentados os principais conceitos relativos a XP que são seus valores, princípios, práticas, papéis e seu ciclo de vida.

3.2 Processos de software

Um processo de software pode ser definido como um conjunto de atividades executadas para desenvolver, dar manutenção e gerenciar sistemas de software [47,51,52,53]. Cada atividade pode ser composta de outras atividades e são realizadas por pessoas, que possuem um determinado papel no processo como: programador, gerente, cliente e outros. Tais pessoas podem utilizar ferramentas e modelos que automatizem e facilitem os seus trabalhos, e à medida que o processo flui, artefatos (código, documentos, modelos e diagramas) são produzidos, atualizados e consumidos nas atividades realizadas.

Ao longo dos anos, muitos padrões de processos de software foram definidos com o objetivo de ajudar as organizações a construírem software de uma forma controlada. Inicialmente, tais padrões tinham o objetivo de ser genéricos no sentido de procurar atender a qualquer tipo de projeto e detalhados no sentido de definir uma grande quantidade de atividades e papéis além de sugerir a elaboração de muitos artefatos e documentos detalhados. Exemplos de processos desse tipo são o Rational Unified Process (RUP) [20] e o OPEN [21] que vêm sendo utilizados com muito sucesso no meio empresarial. Em [54,55] pode ser visto que estes processos são mais adequados a empresas com equipes grandes e projetos complexos onde existe a necessidade de uma extensa e detalhada documentação para redução de riscos.

Recentemente, muitos pesquisadores e especialistas na área de desenvolvimento de software passaram a questionar a eficiência dos processos comentados acima – passando a chamá-los de “processos pesados”. Eles afirmam que os processos pesados “burocratizam” o desenvolvimento de software devido ao excessivo número de atividades e artefatos o que, na opinião deles, acaba desviando o processo daquilo que deveria ser prioritário - a entrega constante de software que “roda” para o cliente.

Esses especialistas fundaram o que veio a se chamar de Aliança Ágil [57] que tem por objetivo encontrar abordagens mais simples e eficientes para o desenvolvimento de software. A idéia desses especialistas foi publicada através de um

documento chamado “Manifesto para o Desenvolvimento Ágil de Software” que valoriza os seguintes itens:

- **Indivíduos e interações** acima de processos e ferramentas;
- **Software funcionado** acima de documentação abrangente;
- **Colaboração com o cliente** acima de negociação de contratos;
- **Responder a mudanças** acima de seguir um plano.

Isso não quer dizer que esse manifesto não considere os itens à direita importantes, mas sim, que os itens à esquerda são considerados de maior valor. Portanto, os processos ágeis, como Extreme Programming (XP) [28,29], DSDM [23], e Crystal [24], focam-se em entrega constante de funcionalidade e interação constante entre os membros da equipe e clientes [19,28,29,54,55]. Estes processos vêm se mostrando eficientes em projetos onde os requisitos são constantemente modificados, os ciclos de desenvolvimento são curtos e as equipes que implementam o projeto são pequenas [54,55,56].

Ambos os grupos de processos (ágeis e pesados) possuem vantagens e desvantagens. Cada grupo pode ser mais adequado para determinados tipos de projeto, dependendo de fatores como: natureza e complexidade da aplicação a ser construída; tamanho e distribuição da equipe; e restrições de prazo e custo impostas pelo cliente.

A descrição detalhada de XP será vista ao longo de todo este capítulo que também fará uma análise dos principais defeitos e virtudes de XP em relação ao desenvolvimento de aplicações Web, bem como apresentará a justificativa pela sua escolha. Na próxima seção serão apresentados alguns conceitos básicos de XP.

3.3 Conceitos básicos de XP

O processo ágil eXtreme Programming (XP) resultou da experiência no projeto C3 Payroll na empresa Chrysler. Este projeto consistia da implementação de um sistema

de folha de pagamento que já havia fracassado anteriormente utilizando outras metodologias. Após o sucesso nesse projeto, XP começou a despontar no meio acadêmico e empresarial e se tornou alvo de inúmeras pesquisas e discussões, além de ser adotado em diversas empresas de software do mundo inteiro e apoiado por grandes "gurus" da orientação a objeto como Kent Beck, Ron Jeffries, Martin Fowler e Grady Booch.

O sucesso e popularidade adquiridos por XP se devem principalmente aos relatos de bons resultados obtidos em projetos, a motivação dos profissionais envolvidos com XP e também devido a sua natureza simples e objetiva por se basear em práticas que já provaram sua eficiência no cenário do desenvolvimento de software. Essas práticas têm como objetivo entregar funcionalidades de forma rápida e eficiente ao cliente. Além disso, XP foi criado considerando que mudanças são inevitáveis e que devem ser incorporadas constantemente. XP se baseia em alguns valores e princípios que serão mostrados na próxima seção.

3.3.1 Valores e Princípios de XP

Extreme Programming (XP) é um processo de desenvolvimento de software que adota os valores de comunicação, simplicidade, feedback e coragem [28,29]. Estes quatro valores servem como critérios que norteiam as pessoas envolvidas no desenvolvimento de software e serão detalhados a seguir.

Comunicação: XP foca em construir um entendimento pessoa-a-pessoa do problema, com o uso mínimo de documentação formal e com o uso máximo de interação "cara-a-cara" entre as pessoas envolvidas no projeto. As práticas de XP como programação em pares, testes e comunicação com o cliente têm o objetivo de estimular a comunicação entre gerentes, programadores e clientes.

Simplicidade: XP sugere que cada membro da equipe adote a solução mais fácil que possa funcionar. O objetivo é fazer aquilo que é mais simples hoje e criar um ambiente em que o custo de mudanças no futuro seja baixo. O objetivo dessa abordagem adotada por XP é evitar a construção antecipada de funcionalidades, como é

feita em muitas metodologias tradicionais, que acabam muitas vezes nem sendo usadas.

Feedback: Os programadores obtêm feedback sobre a lógica dos programas escrevendo e executando casos de teste. Os clientes obtêm feedback através dos testes funcionais criados para todas as estórias (casos de uso simplificados). O feedback é importante, pois possibilita que as pessoas aprendam cada vez mais sobre o sistema e assim corrijam os erros e melhorem o sistema.

Coragem: Ela é necessária para que realmente se aplique XP como deve ser aplicado. Exemplos de atitude que exigem coragem são: alterar código já escrito e que está funcionando; jogar código fora e reescrever tudo de novo; e permitir código compartilhado por todos. Estes exemplos de atitudes podem ser necessários para trazer melhorias ao projeto e não devem ser evitadas simplesmente devido ao medo de tentá-las.

Além dos valores apresentados anteriormente, XP define um conjunto de princípios que devem ser seguidos por equipes que forem usar XP em projetos. Os princípios servirão para ajudar na escolha de alternativas de solução de problemas durante o curso do projeto. Deve-se preferir uma alternativa que atenda aos princípios de uma forma mais completa do que outra que seja incompleta, ou seja, que esteja mais longe da filosofia de XP. Os princípios fundamentais são: feedback rápido, assumir simplicidade, mudança incremental, abraçando mudanças e trabalho de qualidade.

Feedback rápido: A idéia de XP é que os participantes de um projeto como clientes, programadores e gerentes devem estar sempre se comunicando para facilitar o aprendizado coletivo sobre o projeto que está sendo desenvolvido, bem como para alertar rapidamente sobre dúvidas, riscos e problemas de modo a facilitar eventuais ações de contingência.

Assumir simplicidade: Todo problema deve ser tratado para ser resolvido da forma mais simples possível. XP afirma que se deve fazer um bom trabalho (testes, refactoring, comunicação) para resolver hoje os problemas de hoje e confiar na sua habilidade de adicionar complexidade no futuro quando for necessário.

Mudança incremental: Quando muitas mudanças são realizadas todas de uma vez, não se obtém um bom resultado. Em vez disso, esse princípio de XP diz que as mudanças devem ser incrementais e feitas aos poucos. Os programadores têm a liberdade para estar sempre fazendo alterações de melhoria no código e as mudanças de requisitos são incorporadas de forma incremental.

Abraçando mudanças (Embracing Change): XP procura facilitar o ato de incluir alterações através do uso de vários princípios e práticas. A idéia de XP é a de que mudanças devem ser sempre bem vindas independentemente do estágio de evolução do projeto. Isso é muito benéfico em projetos cujos requisitos são bastante voláteis devido aos clientes não saberem o que querem.

Trabalho de qualidade: Em XP, qualidade não é um critério opcional, mas sim obrigatório. Embora a definição de qualidade possa variar de pessoa para pessoa, XP trata qualidade no sentido de se ter um sistema que atenda aos requisitos do cliente, que rode 100% dos casos de teste e que agregue o maior valor possível para o negócio do cliente.

3.4 Visão geral de XP

Esta seção baseia-se nas fontes de informação apresentadas em [28,29], e procura mostrar a estrutura principal e o funcionamento do processo Extreme Programming. A Seção 3.4.1 mostra as práticas que constituem a estrutura de XP. A Seção 3.4.2 mostra o ciclo de vida de um projeto XP. Finalmente, a Seção 3.4.3 descreve a gerência de projetos e os principais papéis envolvidos em um projeto XP.

3.4.1 Práticas de XP

Extreme Programming possui doze práticas que consistem no núcleo principal do processo e que foram criadas com base nos ideais pregados pelos valores e princípios apresentados anteriormente na Seção 3.3.1. Segundo um dos criadores de XP, Kent

Beck, estas práticas não são novidades, mas sim práticas que já vêm sendo utilizadas a muitos anos, com eficiência, em projetos de software. Muitas das práticas de XP não são unanimidades dentro da comunidade de desenvolvimento de software, como por exemplo, programação em pares. No entanto, o valor e benefícios de tais práticas devem ser avaliados em conjunto e não individualmente, pois elas foram criadas para serem usadas coletivamente, de forma a reduzir as fraquezas umas das outras. As doze práticas de XP são comentadas abaixo.

O jogo do planejamento: O planejamento de um release e das iterações são feitos com base nas histórias (casos de uso simplificados) e conta com a colaboração de toda a equipe de desenvolvimento, inclusive o cliente, divididos em dois papéis:

1. **Negócio:** Participam as pessoas que mais entendem sobre o negócio e que possam estabelecer prioridades para as funcionalidades a serem entregues;
2. **Técnico:** Participam as pessoas que irão implementar as funcionalidades descritas. Os técnicos estimam qual o esforço e riscos envolvidos para implementar as funcionalidades e comunicam ao pessoal de negócios.

No final de um release é feita uma determinação rápida do escopo do próximo, através da combinação de estimativas e prioridades do negócio. Um release consiste de várias iterações e, em cada iteração, várias estórias (use case simplificados) são implementadas. Os programadores estimam cada estória e dizem quantas eles podem implementar no final do release. Baseado nesses dados, os clientes escolhem as principais estórias (que agregam maior valor para o negócio) que serão implementadas. As estimativas podem ser refeitas durante as iterações à medida que os programadores aprenderem mais sobre o sistema.

O papel do gerente é muito importante para fazer fluir o jogo do planejamento. Ele tem a responsabilidade de facilitar a coleta de métricas, fazer com que as métricas sejam vistas por aqueles cujo trabalho esteja sendo medido, e ocasionalmente intervir em situações que não podem ser resolvidas de forma distribuída pela equipe.

A ferramenta básica de gerência em XP é a métrica. A métrica básica que é usada durante o jogo do planejamento é a razão entre o tempo estimado para desenvolver e o tempo realmente gasto. Se esta razão aumentar (menos tempo real para um tempo estimado dado) pode significar que a equipe está trabalhando corretamente. Por outro lado, isso também pode significar que a equipe está gastando pouco tempo na codificação e em refatoramento o que poderá prejudicar a qualidade do projeto no longo prazo.

Releases pequenos: Em [28] Beck diz: "cada release deve ser tão pequeno quanto possível, contendo os requisitos mais importantes para o negócio". Isso possibilita ter releases freqüentes o que resulta em maior feedback para clientes e programadores, facilitando o aprendizado e a correção dos defeitos do sistema. Beck sugere que os releases sejam de curta duração, normalmente de dois a três meses.

Metáfora: A intenção da metáfora é oferecer uma visão geral do sistema, em um formato simples, que possa ser compartilhada por clientes e programadores. A idéia da metáfora é que seja feita uma analogia entre o sistema que está sendo desenvolvido e um sistema, não necessariamente de software, que todos entendam, com o objetivo de obter um "vocabulário comum" para a posterior criação de nomes de classes, subsistemas, métodos, etc. Um exemplo de metáfora pode ser visto em [29], onde o sistema C3 payroll de folha de pagamento é descrito como uma linha de montagem onde várias partes referentes às horas trabalhadas vão sendo montadas até construir o contra-cheque de pagamento do funcionário. A metáfora conforme dito em [28] também pode ser vista como uma forma simples de arquitetura do sistema, numa linguagem compreensível tanto por clientes como programadores.

Projeto simples: Pode-se explicar esta prática em duas partes: A primeira diz que devem ser projetadas as funcionalidades que já foram definidas e não as que poderão ser definidas futuramente. A segunda diz que deve ser feito o melhor projeto que possa entregar tais funcionalidades. Esta prática tem o intuito de enfatizar que o projeto simples deve se concentrar em soluções simples e bem estruturadas para os problemas de hoje e que não se deve perder tempo investindo em soluções genéricas que procurem atender a funcionalidades futuras, pois como os requisitos mudam

constantemente tais soluções genéricas podem não ser mais a realidade do futuro. Algumas características de projeto simples citadas em [28,29] são:

- Todos os testes executam com sucesso;
- O projeto expressa a idéia que o programador teve;
- O projeto não possui lógica duplicada.
- O projeto contém o menor número possível de classes e métodos.

Testes constantes: Os testes em XP são feitos antes da programação. Existem dois tipos de teste: teste de unidade e teste funcional. Os testes de unidade são feitos para verificar tudo que possa dar errado. Não é preciso escrever testes de unidade para todos os métodos, conforme mencionado em [28-30]. Os testes unitários são automatizados, e toda vez que o programador escrever código, ele irá verificar se o sistema passa em todos os testes. Os testes funcionais são usados para verificação, junto ao cliente, do sistema como um todo. Os testes servem como um mecanismo para assegurar que o sistema está sempre rodando livre de erros, e também servem para dar feedback aos programadores e clientes sobre as falhas encontradas.

Refatoramento (Refactoring): São constantes melhorias no projeto do software para aumentar sua capacidade de se adaptar a mudanças. O refatoramento consiste em aplicar uma série de passos, como mostrado em [31], para melhorar o projeto do código existente, tornando-o mais simples e melhor estruturado, sem alterar sua funcionalidade.

Programação em pares: Todo o código produzido em XP é escrito por um par de programadores, que possuem papéis distintos, sentados lado-a-lado e olhando para o computador. Um parceiro será responsável pela codificação e pensará nos algoritmos e na lógica de programação. O outro parceiro observa o código produzido e tenta pensar mais estrategicamente em como melhorá-lo e torná-lo mais simples, além de apontar possíveis erros e pontos de falha. Além disso, as duplas são constantemente trocadas e os papéis também com o objetivo de que todos os membros da equipe possam ter conhecimento sobre todas as partes do sistema.

Propriedade coletiva do código: A programação em pares encoraja duas pessoas a trabalharem juntas procurando atingir o melhor resultado possível. A propriedade coletiva encoraja a equipe inteira a trabalhar mais unida em busca de qualidade no código fazendo melhorias e refatoramentos em qualquer parte do código a qualquer tempo. A princípio pode-se pensar que esta prática possa gerar problemas, mas como todos devem respeitar um padrão de codificação e devem realizar todos os testes para verificação de erros esta atividade é feita de uma forma controlada e pode ser facilitada com o uso de uma ferramenta de controle de versão.

Integração contínua: O código das funcionalidades implementadas pode ser integrado várias vezes ao dia. Um modo simples de fazer isso é ter uma máquina dedicada para integração. Quando a máquina estiver livre, um par com código a integrar ocupa a máquina, carrega a versão corrente do sistema, carrega as alterações que fizeram (resolvendo as colisões), e executam os testes até que eles passem (100% corretos). O importante é que na integração as funcionalidades só podem ser integradas se não houver erros, caso contrário os erros devem ser corrigidos.

Semana de quarenta horas: Essa não é uma regra que obriga as equipes em projetos XP a trabalharem somente 40 horas por semana. No entanto, Beck diz que não se deve trabalhar duas semanas seguidas além desse tempo, pois o cansaço e a insatisfação de trabalhar horas extras pode resultar numa queda de qualidade do código.

Cliente no local: Deve ser incluído na equipe uma pessoa da parte do cliente, que irá usar o sistema, para trabalhar junto com os outros e responder as perguntas e dúvidas. Mesmo que não seja possível obter alguém da parte do cliente deve-se ter alguém que tenha conhecimento suficiente do negócio para exercer este papel. O cliente tem um papel importante dentro de um projeto XP já que ele participa do planejamento do projeto escrevendo as histórias e priorizando-as.

Padrões de codificação: Como XP prega a propriedade coletiva de código, onde todos podem alterar e fazer refatoramento de qualquer parte do código a qualquer momento, então é mais do que necessário que se tenha padrões de codificação. O objetivo é que todos programem da mesma forma, facilitando o entendimento do código e as alterações.

3.4.2 Ciclo de vida de um projeto XP

Um projeto XP atravessa algumas fases durante o seu ciclo de vida. Essas fases são compostas de várias tarefas que são executadas. A seguir será dada uma explicação das principais fases de um projeto XP de modo a se ter uma idéia de como o projeto flui ao longo do tempo.

Um projeto XP passa pelas seguintes fases: exploração, planejamento inicial, iterações do release, produção, manutenção e morte.

A fase de exploração é anterior à construção do sistema. Nela, investigações de possíveis soluções são feitas e verifica-se a viabilidade de tais soluções. Os programadores elaboram possíveis arquiteturas e tentam visualizar como o sistema funcionará considerando o ambiente tecnológico (hardware, rede, software, performance, tráfego) onde o sistema irá rodar. Com isso, os programadores e os clientes vão ganhando confiança, e quando eles possuírem estórias suficientes, já poderão começar a construir o primeiro release do sistema. Em [28] sugere-se que seja gasto no máximo duas semanas nessa fase.

A fase de planejamento inicial deve ser usada para que os clientes concordem em uma data para lançamento do primeiro release. O planejamento funciona da seguinte forma: Os programadores, juntamente com o cliente, definem as estórias (use case simplificados) a serem implementadas e as descrevem em cartões. Os programadores assinalam uma certa dificuldade para cada estória e, baseados na sua velocidade de implementação, dizem quantas estórias podem implementar em uma iteração. Depois, os clientes escolhem as estórias de maior valor para serem implementadas na iteração – isso é chamado planejamento de iteração. O processo então se repete até terminar as iterações do release. O tempo para cada iteração deve ser de uma a três semanas e para cada release de dois a quatro meses.

Na fase das iterações do release são escritos os casos de teste funcionais e de unidade. Os programadores vão seguindo mais ou menos o seguinte fluxo de atividades na seguinte ordem (Em cada iteração): escrita dos casos de testes; projeto e refatoramento; codificação; realização dos testes; e integração. À medida que esse fluxo

vai sendo seguido, o sistema vai sendo construído segundo os princípios, valores e práticas apresentados nas seções anteriores. Depois de terminado o primeiro release, já se terá uma idéia melhor das tecnologias e do domínio do problema de modo que as iterações poderão ser mais curtas nos releases subseqüentes e já se podem fazer estimativas mais confiáveis com o que se aprendeu das iterações passadas.

Depois do final do primeiro release, considera-se o início da **fase de produção** onde cada release subseqüente do sistema, depois de construído, é colocado para rodar em um ambiente que simula o ambiente de produção para ver seu comportamento em termos de performance. Pode-se fazer testes de aceitação adicionais para simular o funcionamento real do sistema no ambiente alvo.

A fase de manutenção pode ser considerada como uma característica inerente a um projeto XP. Em XP você está simultaneamente produzindo novas funcionalidades, mantendo o sistema existente funcionando, incorporando novas pessoas na equipe e melhorando o código. Mecanismos como: refatoramento, introdução de novas tecnologias, e introdução de novas idéias de arquitetura podem ser utilizados em um projeto XP. É importante ressaltar que a manutenção dada em um sistema que já está em produção deve ser feita com muita cautela, pois uma alteração errada pode paralisar o funcionamento do sistema resultando em prejuízos para o cliente.

A fase de morte corresponde ao término de um projeto XP. Existem duas razões para se chegar ao final de um projeto, uma boa e a outra ruim. A boa razão é quando o cliente já está satisfeito com o sistema existente e não enxerga nenhuma funcionalidade que possa vir a ser implementada no futuro. A má razão para a morte em XP seria a do projeto ter se tornado economicamente inviável, devido a dificuldades de adicionar funcionalidades a um custo baixo e devido a uma alta taxa de erros.

3.4.3 Papéis envolvidos em XP

A gerência em XP é dividida através de dois papéis: o treinador (coach) e o rastreador (tracker). Esses papéis podem ou não ser executados pela mesma pessoa. O **treinador** se preocupa principalmente com a execução técnica e evolução do processo.

O treinador ideal deve ser um bom comunicador, ter um bom conhecimento técnico, não entrar em pânico e ser confiante. O papel do treinador não é de tomar decisões técnicas, mas de fazer com que todos tomem boas decisões e de facilitar o processo de desenvolvimento.

O rastreamento é outro componente da gerência em XP. O objetivo do **rastreador** (tracker) é coletar métricas sobre o que está sendo desenvolvido e confrontar com as métricas estimadas verificando possíveis divergências. O rastreador deve tomar cuidado para não perturbar muito os programadores pedindo por métricas. Uma forma recomendada é coletar os dados duas vezes por semana.

Além dos papéis gerenciais, apresentados anteriormente, uma equipe que utiliza XP para desenvolver software é composta de outros papéis. Estes são: programador, cliente, testador e consultor.

O programador ocupa o principal papel em XP. Ele analisa, projeta, testa, codifica, e integra o sistema. Além disso, o programador estima a dificuldade das estórias e faz alterações nessas estimativas, caso necessário. Em XP o foco está na programação e o importante é entregar funcionalidades implementadas para o cliente. O programador está sempre escrevendo testes de unidade, codificando e fazendo refatoramento com o objetivo de produzir código de alta qualidade rapidamente.

O cliente escolhe o que vai agregar valor ao seu negócio, escolhe o que deve ser feito primeiro e o que deve ser adiado. Além disso, o cliente define com a ajuda dos testadores, os testes funcionais que irão mostrar se o sistema realmente faz o que deve ser feito.

O testador irá ajudar o cliente na definição e escrita dos testes funcionais. Ele não precisa ser uma pessoa com apenas essa função, pode desempenhar também o papel de programador.

O consultor é necessário apenas em algumas situações onde se precisa de alguém com um elevado nível de conhecimento, por exemplo, um especialista em uma determinada tecnologia sobre determinado assunto que não está sendo bem

compreendido pelas pessoas do projeto. O objetivo da equipe não é obter a solução do consultor para usá-la, mas sim entender o problema e elaborar a sua própria solução.

3.5 Quando não se deve usar XP

Os limites exatos de XP ainda não são conhecidos, mas existem alguns fatores que são fortes indicadores para não usar XP como [28,29]: equipes grandes, clientes desconfiados e tecnologia que não dá suporte facilitado para mudanças. Uma breve lista de possíveis barreiras para o sucesso de um projeto XP são mostradas a seguir.

- **Cultura:** A organização pode estar inserida dentro de uma cultura tradicional de desenvolvimento de software, produzindo muita documentação, gastando muito tempo com análise e projeto antecipado, entre outras coisas. Fazer com que a equipe passe a adotar as práticas de XP e esqueça as antigas pode ser muito difícil;
- **Tamanho da equipe:** Um projeto XP deve possuir uma equipe pequena – geralmente até 12 programadores. É difícil imaginar como ficariam alguns conceitos de XP como comunicação, programação em par e outras em uma equipe grande (Ex. 100 pessoas);
- **Tecnologia:** Não se deve usar XP quando uma tecnologia é complicada para escrever casos de teste, quando retorna feedback em um tempo longo ou quando não incorpora facilmente as mudanças;
- **Espaço físico:** A organização do espaço físico onde a equipe de XP trabalha deve facilitar a comunicação e deixar todos próximos uns dos outros;
- **Cliente:** XP exige que o cliente participe da equipe do projeto e trabalhe no mesmo local dos demais, estando à disposição, de preferência, o tempo todo para esclarecer dúvidas. Caso não possa ser alguém da organização do cliente deve ser alguém que entenda do negócio do

cliente e que seja capacitado para exercer este papel. O cliente também não precisa estar disponível todo o tempo embora seja preferível.

3.6 Considerações finais

Este capítulo apresentou os principais conceitos relativos a Extreme Programming. Foi dada uma explicação sobre a estrutura de XP através da descrição de seus valores, princípios e práticas e também foi mostrado o ciclo de vida que segue um projeto XP. A descrição de XP dada neste capítulo servirá de base para a modelagem mostrada no próximo capítulo.

4

Modelagem de XP usando SPEM

A Seção 4.1 apresenta uma visão geral sobre a modelagem de processos de software. A Seção 4.2 apresenta uma visão geral de SPEM. As seções 4.3 e 4.4 mostram a modelagem de XP usando SPEM. Finalmente, a Seção 4.5 apresenta as considerações finais do capítulo.

4.1 Modelagem de Processos de Software

Um processo de software pode ser definido como um conjunto parcialmente ordenado de atividades realizadas para construir, gerenciar e dar manutenção em sistemas de software [51,52,53]. Tais atividades são realizadas por pessoas que possuem um determinado papel no processo. Durante a execução de atividades, produtos do processo, chamados artefatos, podem ser atualizados, consumidos ou produzidos. Exemplos comuns de artefatos são código fonte, modelos de análise e projeto, diagramas, documentos, manuais e outros.

Pode-se então dizer que um processo de software se compõe de diversos elementos que se relacionam segundo algum critério. Alguns elementos comuns a processos de software segundo [51,52,53] são:

- **Agente ou ator:** É uma entidade que participa da execução do processo. Pode-se dividir os atores em dois grupos: atores humanos e atores de sistema. Os atores humanos são as pessoas que participam do processo enquanto que os atores de sistema são os componentes de hardware e software que participam da execução de uma determinada parte do processo;

- **Papel:** Descreve as responsabilidades, deveres e habilidades necessárias para realizar uma determinada atividade do processo. Um papel pode ser composto por um grupo de agentes e um mesmo agente pode atuar em papéis distintos;
- **Atividade:** Representa o trabalho realizado por um ou mais agentes em um determinado ponto de execução do processo. Os agentes agrupados em papéis realizam uma determinada atividade como responsáveis ou assistentes e podem utilizar técnicas, modelos e ferramentas para facilitar o seu trabalho. Além disso, artefatos podem ser exigidos como entrada para a realização de uma determinada atividade ou podem ser atualizados e produzidos como saída no término da atividade;
- **Artefato ou produto:** É o subproduto de um processo. Ele é criado durante a execução do processo e pode ser modificado ao longo do tempo apresentando diferentes versões. Existem artefatos que são criados para facilitar o processo de construção de software e a manutenção dos sistemas (ex: modelos de dados, diagrama de classes e outros). Nem todo artefato é entregue ao cliente, ficando a critério da empresa fornecedora de software quais artefatos devem ou não ser descartados ao longo do tempo.

A descrição dos elementos do processo pode ser representada através de um diagrama de classes da UML como na Figura 4.1.

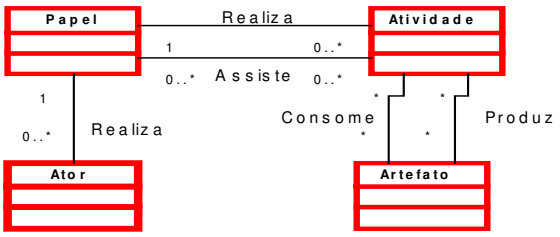


Figura 4.1 - Elementos de um processo de software

A modelagem de processos de software procura definir modelos e seus inter-relacionamentos para representar os elementos que fazem parte do processo, como, por

exemplo, na Figura 4.1. Tais modelos podem ser construídos usando-se uma determinada linguagem de modelagem e procuram retratar uma determinada visão do processo de software. Existem diversas linguagens, meta-modelos e ferramentas que possibilitam modelar diversos aspectos de processos de software segundo uma visão particular. Nas próximas seções será discutido a importância da modelagem de processos de software (Seção 4.1.1) bem como algumas abordagens existentes para modelagem (Seção 4.1.2).

4.1.1 Importância da modelagem de processos de software

Alguns trabalhos [51,52] têm constatado os benefícios e a importância da modelagem de processos de software. Dentre os principais benefícios advindos da modelagem pode-se citar:

- **Facilitar o entendimento e comunicação:** A modelagem do processo permite a estruturação dos seus elementos mostrando seus inter-relacionamentos e tornando mais fácil a visualização e entendimento do processo por parte das pessoas envolvidas;
- **Facilitar a gerência e controle do processo:** A modelagem pode simplificar o trabalho da gerência, que poderá verificar, de uma forma mais fácil – através de uma determinada visão do processo - o andamento do processo dentro de uma seqüência de atividades a ser seguida;
- **Dar suporte para a melhoria do processo:** Uma vez que se tenha um modelo para o processo, fica mais simples evoluir e fazer modificações para atingir determinados objetivos específicos e adaptar o processo a novos conceitos e paradigmas. Ao se acostumar com o processo na forma de um modelo, as pessoas tendem a entender cada vez mais o processo e sugerir adaptações e melhorias ao longo do tempo;
- **Dar suporte para a automação de partes do processo:** Uma vez que se tenha definido um modelo para representar processos, ferramentas

> podem ser criadas para automatizar certas partes do processo, eliminando algumas atividades feitas de forma manual e, conseqüentemente, agilizando o processo de construção de software.

Levando esses benefícios em consideração é que este trabalho procura elaborar uma modelagem para o processo de software XP. Em [65] menciona-se que processos de software ágeis, incluindo XP, carecem de uma descrição mais explícita dos seus elementos e não especificam como podem ser adaptados. Isso torna difícil a adaptação de processos ágeis para determinados tipos de propósitos como, por exemplo: adaptar Extreme Programming para o desenvolvimento de software para a Web ou adaptar XP para o desenvolvimento de software de tempo real.

A necessidade de se adaptar processos surge do fato de que os processos de desenvolvimento de software geralmente possuem um propósito geral e são descritos de uma maneira que não abrange de forma suficiente todos os tipos de domínios de problema. Então, para resolver um determinado problema específico, em alguns casos, é necessária uma adaptação no processo de software. A tarefa de adaptar o processo pode ser facilitada pelo uso linguagens de modelagem para representar o processo e ferramentas para automatizar a modelagem.

Ao longo deste capítulo serão mostradas a modelagem e as adaptações feitas em XP. A modelagem é feita usando a meta-linguagem de modelagem de processos SPEM [48] e será descrita nas seções 4.3 e 4.4. Esta modelagem servirá para a elaboração do processo XWebProcess a ser mostrada no Capítulo 5.

4.1.2 Abordagens para modelagem de processos de software

Processos de software podem ser modelados em diferentes níveis de abstração como, por exemplo: modelos genéricos, modelos adaptados e modelos instanciados. Além disso, eles também podem ser modelados com diferentes objetivos e segundo visões distintas. Em [66] são descritas algumas das principais perspectivas da modelagem de processos:

- **Funcional:** Representa quais elementos do processo estão sendo implementados e quais fluxos de informação são importantes para tais elementos;
- **Comportamental:** Representa quando e sobre quais condições os elementos do processo são implementados;
- **Informativo ou Estrutural:** Representa a estrutura e relacionamentos entre os elementos do processo.

Os modelos usados para representar processos procuram levar em consideração as perspectivas vistas acima e são criados com o objetivo de se oferecer linguagens, abstrações e formalismos para facilitar a representação de um processo de software capturando sua estrutura e comportamento.

Ao longo dos últimos anos, surgiram diversas linguagens com o propósito de modelar processos de software. Em [67] é mostrada uma linguagem chamada DYNAMITE que se baseia em conceitos de UML e de redes de tarefas (task nets) para a modelagem de processos de software. Além da linguagem, ferramentas são fornecidas para criar os modelos e acompanhar o processo à medida que um projeto vai sendo implementado. Em [68] é proposta uma linguagem orientada a objetos para modelagem de processos de software que também possui o suporte de ferramentas.

O surgimento de várias linguagens para processos de software acaba acarretando um problema semelhante ao que aconteceu nos anos 80 em relação à modelagem orientada a objetos, quando muitas empresas usavam linguagens distintas para modelagem o que dificultava o trabalho junto aos clientes e a troca de informações entre as empresas. Esse problema foi minimizado depois do surgimento de UML que logo veio a se tornar um padrão utilizado em escala mundial.

No campo da modelagem de processos de software um grande avanço foi dado nos últimos três anos quando a OMG [69] decidiu submeter uma requisição (RFP) por uma linguagem de modelagem para processos de software. Diversas empresas importantes na área de software como Rational software, IBM, Unisys, Alcatel e outras resolveram se unir para elaborar uma linguagem que pudesse atender aos requisitos

propostos pela OMG. Em Novembro de 2002 a meta-linguagem Software Process Engineering Metamodel (SPEM) [48] foi oficializada como um padrão da OMG e encontra-se atualmente na sua versão 1.0.

A meta-linguagem SPEM foi escolhida como linguagem de modelagem a ser usada neste trabalho, pois acreditamos que ela possa se tornar um padrão muito utilizado, como é a UML, devido aos seguintes fatores:

- É um padrão oficial da OMG cuja reputação em estabelecer padrões é indiscutível;
- Possui o apoio de grandes empresas na área de software que estão dando suporte ao seu uso, através da construção de ferramentas e outras formas de apoio;
- Baseia-se em UML que é um padrão já conhecido e consolidado no meio acadêmico e empresarial.

Uma descrição completa da linguagem SPEM é mostrada no Apêndice 1, onde é apresentada uma visão geral sobre a linguagem e seus principais conceitos. A próxima seção mostra uma visão geral de SPEM.

4.2 Visão geral de SPEM

O Software Process Engineering Metamodel (SPEM) [48] é um meta-modelo que pode ser usado para descrever um processo concreto de desenvolvimento de software ou uma família de processos relacionados. SPEM adota uma abordagem orientada a objetos para modelar processos e usa UML como notação. A Figura 4.2 descreve a arquitetura de quatro níveis de modelagem definida pela OMG e respeitada por SPEM.

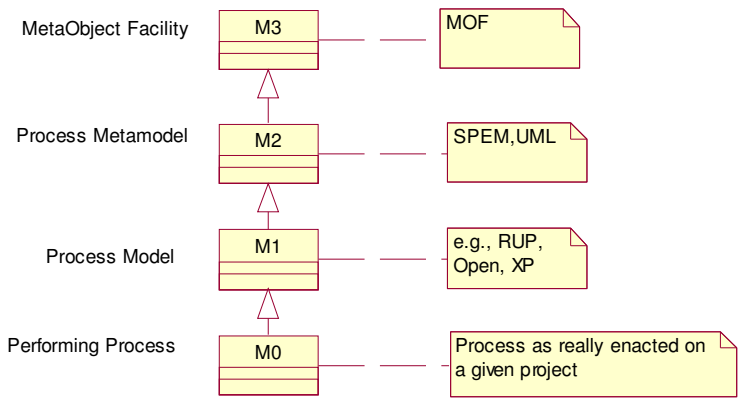


Figura 4.2 - Níveis de modelagem definidos em [48]

O processo real de produção, conforme instanciado para um projeto específico, está no nível M0. No nível M1 encontra-se a definição do processo que foi instanciado em M0 como, por exemplo, RUP, OPEN ou XP. O foco da definição de SPEM está em criar um meta-modelo que se encontra no nível M2 que possui estereótipos para a modelagem dos processos que estão no nível M1. A especificação de SPEM está estruturada como um perfil UML (UML profile)[6] e também através de um meta-modelo baseado no MOF[7] que se encontra no nível M3. Essa abordagem facilita a utilização e construção de ferramentas para UML e ferramentas baseadas em MOF.

O objetivo do padrão SPEM é abranger uma vasta gama de processos de desenvolvimento de software já existentes ao invés de excluí-los devido ao excesso de elementos e restrições que poderiam existir na sua definição. SPEM permite que seus usuários (modeladores de processo) usem a UML, e define estereótipos que podem ser usados nos modelos produzidos.

O meta-modelo SPEM é construído pela extensão de um subconjunto do meta-modelo da UML 1.4. Esse subconjunto de UML é denominado na definição de SPEM como “SPEM_Foundation”. Além disso, o modelo SPEM possui o pacote “SPEM

---

[6] Um UML-profile é um conjunto de uma ou mais extensões da semântica de UML com a intenção de customizá-la para um domínio ou propósito particular, como, por exemplo, para modelagem de processos no caso de SPEM.

[7] O Meta-Object Facility (MOF) é a tecnologia adotada pela OMG para definir metadados e representá-los como objetos CORBA. SPEM usa um subconjunto da UML para representar seus elementos como um meta-modelo MOF.

Extensions" que adiciona as construções e semânticas requeridas para a engenharia de processos de software. A Figura 4.3 abaixo mostra esses dois pacotes.

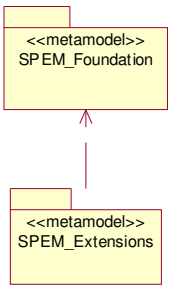


Figura 4.3 - Estrutura de pacotes de SPEM [48]

Um maior detalhamento do conteúdo de cada pacote pode ser encontrado em [48] e no Apêndice 1. A descrição do modelo SPEM apresentada até agora é suficiente para se ter um entendimento de como a linguagem foi construída e de qual é a sua utilidade. O uso real de SPEM para a modelagem de um processo de software será apresentado nas duas próximas seções através da modelagem de XP.

4.3 Modelagem dinâmica de XP usando SPEM

Na Seção 4.2 foi apresentada uma visão geral de SPEM destacando-se como a linguagem é definida e organizada. Nesta seção, procura-se mostrar como SPEM pode ser utilizada na prática, ou seja, do ponto de vista de seus usuários. A linguagem SPEM será utilizada para modelar o processo XP, apresentado anteriormente, sob as perspectivas dinâmica (comportamental) e estrutural.

De acordo com [48] alguns diagramas básicos de UML podem ser usados para apresentar perspectivas diferentes de um modelo de processo de software. Em SPEM podem ser utilizados alguns desses diagramas, dentre eles: diagrama de classes, de pacotes, de atividade, de caso de uso, de seqüência e de transição de estados. A linguagem SPEM oferece algumas representações e estereótipos para modelar seus

principais elementos em diagramas UML. Um resumo dos principais elementos de SPEM, seus conceitos e suas representações gráficas, é mostrado na Tabela 4.1.

Tabela 4.1 - Descrição de alguns elementos de SPEM[8]

ESTEREÓTIPO	COMENTÁRIO	NOTAÇÃO
WorkProduct (Artefato)	É uma descrição de algo que contém informação ou é uma entidade física produzida ou usada por atividades do processo. Ex: modelos, planos e documentos.	
WorkDefinition (Conjunto de Trabalho)	É um elemento do modelo que descreve a execução, as operações realizadas e as transformações feitas nos "WorkProducts". Representa um conjunto de atividades, como, por exemplo: especificar requisitos e fazer projeto do sistema.	
Guidance (Guia)	É um elemento do modelo que se associa a outros elementos e pode conter descrições adicionais, tais como técnicas, "guidelines" e "templates".	
Activity (Atividade)	É uma "WorkDefinition" que descreve o que um "ProcessRole" (papel) realiza. A diferença para "WorkDefinition" é sutil mas pode ser entendida como sendo uma única atividade bem delimitada, e não um conjunto, de responsabilidade de um determinado papel.	
ProcessRole (Papel)	Descreve os papéis, responsabilidades e competências que um determinado indivíduo tem dentro do processo.	

[8] Estes estereótipos são definidos em [48] e no restante do trabalho serão usados em português.

ESTEREÓTIPO	COMENTÁRIO	NOTAÇÃO
Discipline (Disciplina)	É um agrupamento coerente de elementos do processo (artefatos, papéis, atividades) cujas atividades são organizadas segundo algum ponto de vista ou tema comum (Ex: Análise e Projeto, teste, implementação, etc.).	

Os elementos descritos na Tabela 4.1 podem ser usados em diagramas UML para representar diversos pontos de vista do processo de software. Neste trabalho, será mostrada a modelagem feita para o processo Extreme Programming utilizando os conceitos acima. A Figura 4.4 mostra um diagrama de atividades que representa a dinâmica de um projeto realizado seguindo o processo XP.

O processo se inicia por uma etapa de explorações, onde são feitos testes e experimentos com as tecnologias a serem utilizadas e verifica-se a existência de viabilidade tecnológica considerando-se as restrições do ambiente onde o sistema será implantado. Uma arquitetura inicial e um levantamento prévio dos requisitos do sistema são elaborados.

Depois disso, os clientes e os programadores escrevem as estórias que representam os requisitos do release a ser implementado. As estórias correspondem a casos de uso simplificados que recebem uma estimativa de dificuldade dada pelos programadores com base em estimativas passadas e na sua experiência. Essas estimativas servem como uma base para que os clientes escolham quantas histórias serão implementadas no release – isso segue a idéia do jogo do planejamento apresentado na Seção 3.4.1.

Dentro de um release, ocorrem várias iterações onde diversas histórias são implementadas e testadas. Cada iteração segue um determinado fluxo de atividades que consistem em: planejar iteração, projetar, escrever testes de unidade, codificar, testar e integrar conforme mostrado na Figura 4.4. Além disso, dentro da iteração podem ocorrer diversas alterações nos requisitos onde as estórias podem ser re-escritas e revisadas para atender as necessidades de mudança do cliente. Essas alterações podem

ser realizadas durante o planejamento da iteração, onde os programadores podem reavaliar as estimativas feitas anteriormente com base nas sugestões de modificação e no feedback obtido de iterações anteriores.

Ao término da última iteração, a versão atual do sistema é colocada em produção no ambiente do cliente e diversas versões subseqüentes são geradas até que todos os requisitos do sistema sejam implementados e entregues ao cliente.

A modelagem dinâmica de XP mostrada na Figura 4.4 foi feita com base nos estereótipos de SPEM utilizando-se um diagrama de atividades da UML para dar uma idéia da seqüência de cada disciplina de XP ao longo do tempo. A modelagem se baseou na descrição do ciclo de vida de XP apresentada na Seção 3.4.2. A próxima seção procura apresentar a modelagem estática de algumas disciplinas através de diagramas de classe UML.

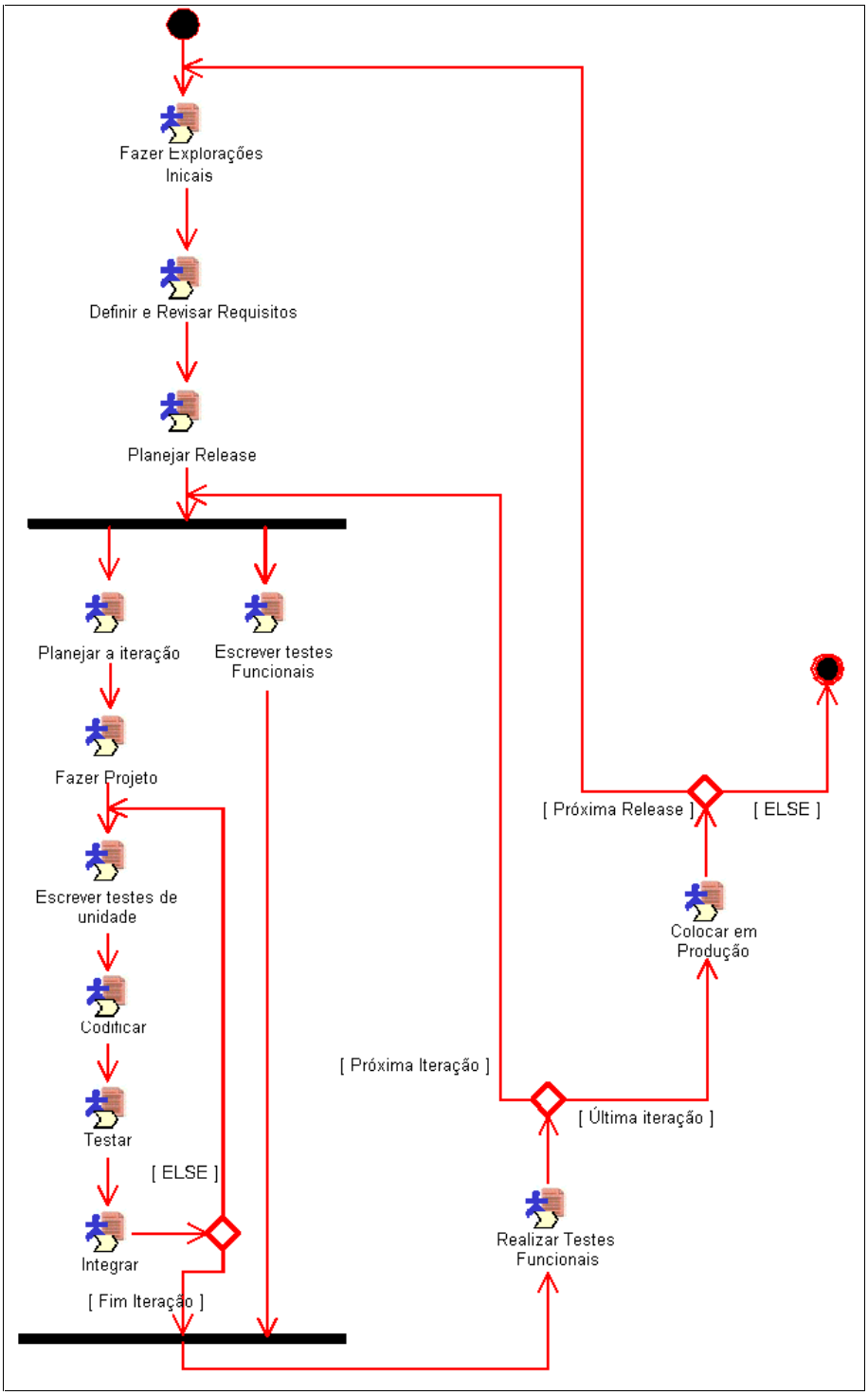


Figura 4.4 - Modelagem de XP usando diagrama de atividades da UML e estereótipos de SPEM

4.4 Modelagem estática das disciplinas de XP

Nesta seção são apresentados diagramas de classe que descrevem o detalhamento de duas disciplinas da Figura 4.4. Estas duas disciplinas são mostradas na Figura 4.5 e na Figura 4.6 e representam, respectivamente, as disciplinas Fazer explorações iniciais e Definir e revisar requisitos. A apresentação completa de todos os diagramas é feita no Apêndice B.

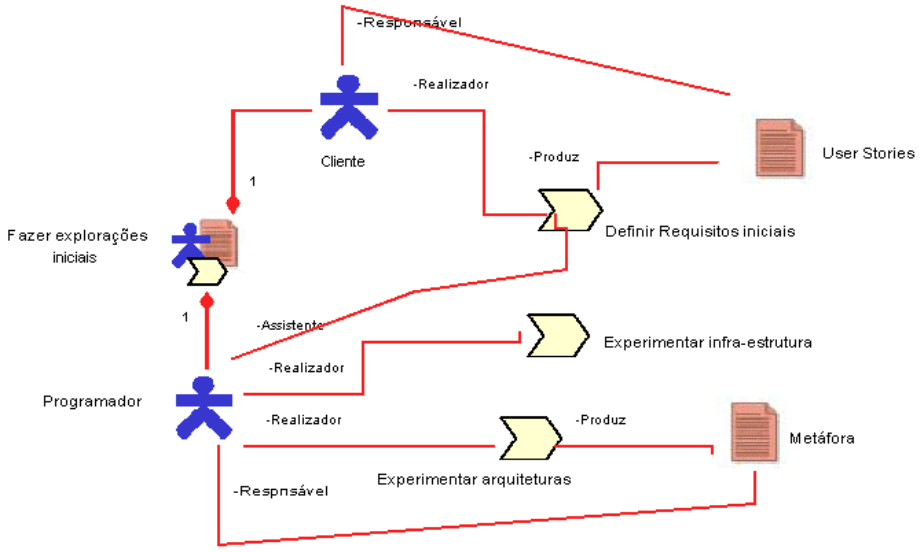


Figura 4.5 - Fazer explorações iniciais

A modelagem da Figura 4.5 procura representar o que é feito na fase de exploração de XP descrita na Seção 3.4.2. Esta parte do processo XP é onde são feitas as primeiras explorações e investigações sobre o projeto a ser implementado. Os clientes são consultados e trazidos para dentro da equipe para realizar a definição prévia dos requisitos e do escopo do sistema. Os programadores são responsáveis por fazer experimentos com a possível tecnologia e infra-estrutura a ser escolhida para verificar a viabilidade da solução e elaborar uma idéia de arquitetura através de uma metáfora. Algumas estórias iniciais podem ser descritas, mas o importante aqui é levantar as informações necessárias para decidir se o projeto é viável ou não. Outra disciplina de XP é descrita na Figura 4.6.

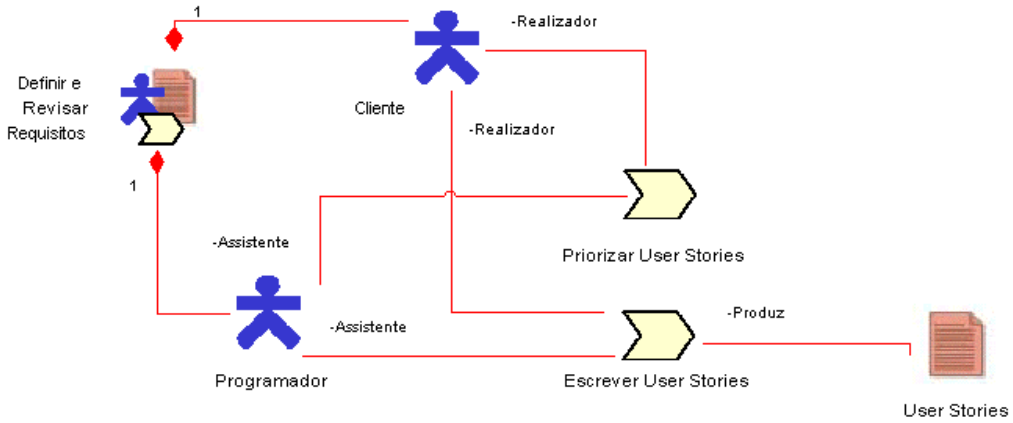


Figura 4.6 - Definir e Revisar requisitos

A modelagem da Figura 4.6 procura representar o que é descrito nas seções 3.4.1 e 3.4.2 no que se refere ao levantamento de requisitos, que é feito basicamente pelo cliente e pelos programadores, através da escrita dos cartões de história.

Esta parte do processo XP se preocupa em definir e revisar os requisitos que irão fazer parte de uma versão (release) do sistema. Como XP admite mudanças constantes nos requisitos, então é importante mencionar que esses requisitos podem ser alterados a qualquer instante e que eles também serão revistos na parte de XP que trata dos requisitos da iteração. Em XP, o cliente, ou alguém que o represente, trabalha juntamente com a equipe de desenvolvimento (prática on-site customer) e é responsável por escrever e priorizar as histórias com o auxílio dos programadores. Os clientes escrevem os cartões contendo as descrições de cada estória, as quais são atribuídas uma certa prioridade pelo cliente. Os principais resultados dessa disciplina são os cartões de estória com a descrição de suas funcionalidades e as estimativas de esforço.

Ao longo desta seção, a meta-linguagem de modelagem SPEM foi usada para modelar a estrutura do processo de software XP, através da elaboração de dois diagramas de classe contendo estereótipos de SPEM. Alguns dos benefícios da modelagem de processos já foram citados na Seção 4.1.1 e a justificativa da escolha de SPEM foi mencionada na Seção 4.1.2. É importante também ressaltar que o fato de se ter modelado XP não implica em nenhuma perda de eficiência, pois a modelagem não

tem o objetivo de alterar a forma como se trabalha utilizando XP, nem tampouco sugere nenhuma modificação na sua estrutura.

A modelagem de XP realizada traz os seguintes benefícios:

- Simplifica o entendimento dos elementos de XP tornando mais fácil seu aprendizado;
- Simplifica o trabalho de uma organização que pretende adotar XP como processo base de engenharia de software;
- Simplifica o trabalho de uma organização que já utiliza XP e precisa fazer adaptações para atender certas características de projetos como, por exemplo, desenvolvimento de software para a Web e de software de tempo real usando XP;
- A forma de trabalhar com XP não é alterada pelo modelo, logo XP não deixará de ser ágil ou leve porque foi modelado.

A modelagem dinâmica e estática de XP produzidas nesta seção e na anterior serve como base para as adaptações feitas no próximo capítulo. A organização e a estruturação de XP em modelos facilitam o seu entendimento e também a elaboração do processo XWebProcess.

4.5 Considerações finais

Este capítulo apresentou uma parte da modelagem de XP usando SPEM. O principal objetivo de elaborar a modelagem foi facilitar o entendimento e a organização dos elementos de XP. Esta modelagem servirá também para facilitar a criação do processo XWebProcess a ser apresentado no próximo capítulo. Alguns dos benefícios advindos desta modelagem serão apontados no experimento realizado no Capítulo 6.

5
O processo XWebProcess

Este capítulo apresenta o processo XWebProcess. A Seção 5.1 apresenta uma visão geral sobre XWebProcess. A Seção 5.2 contextualiza XP sob a ótica do desenvolvimento Web e a Seção 5.3 explica em que se baseou a elaboração do processo. A Seção 5.4 mostra o processo XWebProcess e a Seção 5.5 explica como aplicar o processo em projetos. O capítulo se encerra com as considerações finais mostradas na Seção 5.6.

5.1 Visão geral

Os capítulos anteriores apresentaram os conceitos básicos relativos à engenharia de software para a Web e sobre o processo ágil Extreme Programming. Estes conceitos apresentados servem de base para o entendimento de XWebProcess [84], um processo ágil de desenvolvimento de aplicações Web, que é apresentado neste capítulo.

Conforme discutido anteriormente, o software dentro das organizações passou a ter um papel estratégico. É cada vez mais comum o número de empresas que passam a depender de sistemas não só como mecanismos operacionais para automatizar processos de negócio, mas também como elemento estratégico que permite oferecer novos serviços e produtos para ganhar vantagens competitivas.

Dentro desse contexto, ao longo dos últimos anos, os processos ágeis como XP vêm ganhando bastante atenção por parte da indústria e dos centros de pesquisa. Devido à necessidade de construir software de forma rápida e eficiente para atender a demanda dos clientes, os processos ágeis passam a ser uma boa alternativa.

No caso das aplicações Web, as questões ligadas a restrições de tempo ficam ainda mais evidentes. Expressões criadas no mundo dos negócios como, por exemplo,

"Web Time" são evidências de quanto essas restrições de tempo são importantes nessas aplicações. Atualmente, existem empresas que oferecem e vendem seus serviços totalmente pela Web como, por exemplo, "Amazon.com" e "google". Não seria exagero nesses casos dizer então que a aplicação Web é a empresa. Tais empresas conquistaram uma grande fatia de mercado devido principalmente a serem pioneiras nas suas idéias e na capacidade de implementá-las antes dos concorrentes.

Portanto, considerando o caso de aplicações Web devem existir processos de desenvolvimento de software capazes de lidar com as restrições de tempo impostas pela sua natureza estratégica e competitiva. Estes processos devem procurar não só construir as aplicações de forma rápida, mas também de forma adequada e com qualidade. É nesse contexto que surge o processo XWebProcess que tem como objetivos:

- Ser um processo ágil para desenvolvimento de aplicações Web que respeite os princípios e valores de XP e também siga a filosofia pregada pela Aliança Ágil [57];
- Ser um processo independente de métodos, técnicas, ferramentas e plataformas de desenvolvimento. Ou seja, em XWebProcess não importa, por exemplo, se alguém usa a plataforma .NET ou J2EE para o desenvolvimento da aplicação. Isso será mais detalhado na Seção 5.5.1;
- Ser um processo que tenha uma definição clara dos seus elementos (disciplinas, papéis e artefatos) que seja de fácil compreensão para equipes que queiram utilizá-lo;
- Ser um processo que possa ser facilmente adaptado para atender a restrições e características não previstas na sua elaboração, ou seja, XWebProcess pode ser estendido.

Além dos objetivos citados anteriormente, XWebProcess possui algumas restrições que devem ser observadas. São elas:

- A concepção de XWebProcess considerou as características gerais do desenvolvimento Web, sendo assim, podem existir certos tipos de aplicações Web que necessitem de adaptações no processo para atender a

condições que não foram previstas. Isso será mais detalhado na Seção 5.5.1;

- Qualquer extensão ou adaptação feita em XWebProcess deve respeitar os princípios e valores dos processos ágeis e as guias estabelecidas na Seção 5.5.2. O Objetivo disto é que as extensões feitas em XWebProcess não devem torná-lo um processo muito pesado o que iria contra a filosofia do processo.

Portanto, os principais objetivos e restrições relativos a XWebProcess servem para dar uma boa idéia do que o processo procura ou não abordar. Na próxima seção será feita uma contextualização de XP em relação ao desenvolvimento Web que servirá de base para a concepção de XWebProcess.

5.2 Contextualização de XP para o desenvolvimento Web

Esta seção tem por objetivo justificar a escolha de XP como base para o processo XWebProcess (Seção 5.2.1) além de fazer uma análise das deficiências e virtudes de XP no contexto do desenvolvimento Web (Seção 5.2.2), o que permitirá levantar as necessidades de adaptação de XP (Seção 5.2.3) e finalmente, discutir sobre extensões de XP (Seção 5.2.4).

5.2.1 Justificativa pela escolha de XP

Ao longo dos últimos anos, dados coletados sobre o uso de XP em projetos vêm mostrando sua eficiência e comprovando a sua crescente popularidade. Em [58] é apresentada uma pesquisa mostrando dados quantitativos sobre projetos XP que podem ajudar a justificar a utilização de XP no desenvolvimento Web.

Essa pesquisa se baseou em 45 questionários enviados por e-mail para listas de discussão e empresas de diversos setores (software, telecomunicações, financeiro,

consultoria, biotecnologia e outros) ao redor do mundo (Europa, América do Norte, e Ásia). Além disso, as empresas variavam quanto ao seu tamanho indo desde pequenas empresas até multinacionais. Dados sobre os percentuais de cada ramo de negócio, localização e tamanho das empresas pesquisadas podem ser encontrados em [58]. Quanto ao tipo de aplicação, 28,9% dos projetos correspondiam a software para a Web.

Alguns resultados obtidos pela pesquisa comprovam a eficiência de XP em projetos de software em geral - incluindo projetos Web como visto acima. Os resultados mais interessantes são:

- Quase todos os projetos foram bem sucedidos, e uma parte significativa desses projetos consistia de aplicações Web (28,9%);
- 100% dos desenvolvedores, independentemente do tipo de aplicação que desenvolvem, usariam XP novamente em um próximo projeto, se apropriado;
- Os problemas em usar XP normalmente se referem a preconceitos como: gerentes céticos, desenvolvedores que se recusam a programar em par, e a filosofia da empresa não permite cliente no local (on-site customer).

Portanto, essa pesquisa consegue apresentar evidências quantitativas, obtidas através da análise de dados coletados de projetos reais, que Extreme Programming pode ser usado com sucesso em projetos Web.

Além disso, em [59,60] comenta-se que as práticas e princípios de XP podem ajudar a minimizar riscos ligados ao desenvolvimento de software para Web como:

- Resposta eficiente a mudanças no negócio. XP através de práticas como refatoramento, projeto simples, testes automatizados e “releases” pequenas cria um ambiente que acolhe mudanças constantes a um baixo custo;
- Melhora a produtividade ao longo do ciclo de vida de software. Isso é um objetivo que é comum a todos os processos ágeis, ou seja, estar

constantemente entregando valor para o cliente em termos de software rodando;

- O envolvimento do cliente permite ajudar a esclarecer o escopo do projeto o que é fundamental em projetos Web, cujos clientes normalmente não sabem o que querem inicialmente.

Os fatos apresentados anteriormente indicam que XP pode ser um processo utilizado para o desenvolvimento de software para a Web. Somando-se a isso pode ser acrescentado o destaque que este processo vem ganhando no cenário mundial, tanto no meio acadêmico quanto industrial. Obviamente, XP possui alguns problemas que precisam ser tratados e que são assunto das duas próximas seções.

5.2.2 Defeitos e virtudes de XP para o desenvolvimento Web

No Capítulo 2 foi comentado que muitos projetos Web possuem características como: equipes multidisciplinares; ciclos curtos de desenvolvimento; grande quantidade de requisitos não funcionais; alteração constante nos requisitos; preocupação com questões de navegação e interfaces; e preocupação com as questões relativas à evolução e integração dos sistemas Web.

Diante disso, percebemos que XP trata adequadamente algumas dessas características e falha em outras. Algumas das falhas que podem ser apontadas em XP se referem principalmente às seguintes características [4]:

- **Projeto de navegação e interface:** Como XP é um processo de software genérico que pode servir para qualquer tipo de aplicação, esta preocupação característica de sistemas Web não é tratada. Uma das questões fundamentais no desenvolvimento de sistemas para a Web é a do projeto da navegação e das páginas do Website. Os sistemas Web são concebidos para serem usados por diversos tipos de usuário, com perfis e funções diferentes, que podem navegar de diversas maneiras para obter informações de seu interesse. Então, um processo que trate da construção de sistemas Web deve levar isso em consideração. Para

solucionar este problema pode-se pensar em importar para XP, práticas de projeto de navegação e interface adotadas por outras metodologias como as técnicas apresentadas na Seção 2.4. É importante mencionar que a incorporação de tais práticas deve ser feita respeitando os valores e princípios de XP, caso contrário ela pode perder a sua característica de processo ágil;

- **Arquitetura:** Como visto anteriormente na definição de XP, não existe uma preocupação em projetar uma arquitetura para o sistema. A metáfora, que é uma espécie de arquitetura simplificada, pode não ser adequada para sistemas Web onde a arquitetura tem um papel muito importante, pois deve ser flexível e bem projetada para facilitar mudanças. As questões arquiteturais em XP são descritas de forma superficial através da metáfora. Esta serve como uma estória que pode ser compartilhada pelas pessoas envolvidas no projeto para dar uma visão geral dos elementos do sistema. O problema da metáfora é que ela é relativamente estática e estável o que não se alinha ao desenvolvimento Web que deve possuir um alto grau de dinamismo para incorporar as mudanças de infraestrutura e permitir a evolução dos sistemas Web. É importante então, dar uma maior ênfase às questões arquiteturais, que são de fundamental importância para o funcionamento, evolução e integração dos sistemas Web ao longo do tempo.

As virtudes de XP são:

- **Simplicidade:** Existe uma preocupação em construir soluções simples dentro de um projeto de código bem feito que pode incorporar mudanças de forma pouco custosa;
- **Tamanho da equipe:** Equipes pequenas que procuram entregar constantemente as funcionalidades de maior valor do negócio para o cliente. Conforma citado em [4] a maioria dos projetos Web são desenvolvidos por equipes pequenas;

- **Releases pequenos e freqüentes:** As freqüentes entregas de funcionalidades que agregam o maior valor possível para o cliente ajudam a reduzir riscos e erros dos sistemas. Isso é fundamental em sistemas Web onde os requisitos costumam mudar bastante e com freqüência.

5.2.3 Necessidades de adaptação de XP

Com base na análise da Seção 3.4 onde foi mostrada uma visão geral de XP e nas duas seções anteriores (5.2.1 e 5.2.2), esta Seção procura dar uma idéia de quais pontos de XP precisam de adaptações. A idéia aqui será comentar quais as necessidades de adaptação considerando cada disciplina do processo de referência mostrado anteriormente na Seção 2.7. Isso é apresentado na Tabela 5.1.

Tabela 5.1 - Necessidades de adaptação de XP

DISCIPLINA	DESCRIÇÃO
Análise de domínio	Essencialmente, o objetivo da análise de domínio é entender claramente os problemas a serem resolvidos pela aplicação Web. Antes de começar a escrever qualquer código é preciso estudar a viabilidade do projeto e saber se realmente o sistema Web trará benefícios econômicos para a organização. Tais benefícios podem ser: redução de custos com a automatização de processos de negócio; oferta de novos serviços e produtos; aumento do número de clientes; e ganhar vantagens competitivas. Para entender os problemas a serem abordados a equipe pode analisar soluções similares construídas por concorrentes e pedir a opinião de especialistas como consultores. Então, durante essa fase deve-se tentar entender os processos de negócio no qual o sistema Web será inserido. Tais processos também podem passar por adaptações e por reengenharia para se adaptar a realidade da Web. A fase de exploração de XP, explicada em [28,29], pode então ser moldada para essa tarefa de entender o modelo de negócio. Fica a critério da equipe de desenvolvimento a construção ou não de um documento formal do modelo de negócios – embora XP não seja favorável a muita documentação.
Especificação de requisitos	Após a equipe ter um entendimento e concordar com os problemas que devem ser resolvidos, a análise de requisitos pode iniciar. Esta fase procura definir o que o sistema fará para resolver o problema

DISCIPLINA	DESCRIÇÃO
	(requisitos funcionais) e as restrições inerentes ao problema (requisitos não funcionais). Para fazer um levantamento dos requisitos podem ser utilizadas várias técnicas como: cartões de estória (story cards); diagramas Use Case; e cartões CRC. Além disso, a construção de um protótipo inicial do sistema também pode ser considerada, pois ajuda a entender melhor o sistema e, no caso de sistemas Web, é muito útil para decidir questões de navegação e uso do sistema. Anteriormente foi visto que os requisitos não funcionais possuem grande importância para o desenvolvimento Web. Questões como concorrência, distribuição, tráfego na rede, segurança, número de usuários acessando o sistema e outras são muito importantes para o sucesso de uma aplicação Web. Uma atenção maior para esses assuntos deve ser considerada em XP quando adaptada para o desenvolvimento Web.
Análise e Projeto	O principal objetivo dessa disciplina é representar as principais questões do sistema através de modelos que sejam independentes de implementação. Existem diversas técnicas que podem dar suporte para a análise e projeto de sistemas Web como, por exemplo, extensões de UML (WAE) [12] e OOHDM [18] para projetar a navegação e a interface. Como foi apontada na Seção 5.2.2 existe uma falha em XP que se refere à ausência de técnicas para modelar navegação e interface. XP também não recomenda a utilização de muita modelagem, pois seu principal objetivo é o de codificar a maior parte do tempo possível. Apesar disso, O uso de técnicas ágeis de modelagem em um processo baseado em XP, como sugerido em [34], pode ser bastante eficiente. Neste sentido, uma equipe que utilize XP pode pensar em incorporar técnicas e ferramentas para modelagem desde que respeitem os valores e princípios de XP para não torná-la um processo pesado.
Implementação	Na disciplina de implementação algumas questões devem ser consideradas. A implementação da aplicação em si e a construção das páginas HTML e da implementação da camada de dados podem ser realizadas em paralelo. Estas atividades são de responsabilidades de pessoas com papéis diferentes na equipe – programadores, designers gráficos e projetistas de dados. Aqui, não há muito a ser discutido em relação à adaptação de XP. O processo segue suas práticas normalmente como programação em pares, refatoramento e outras. A única questão que deve ser levada em consideração é como fazer a implementação das páginas Web e do banco de dados e fazer elas se comunicarem com os objetos de negócio, no entanto isso é uma questão arquitetural.

DISCIPLINA	DESCRIÇÃO
Testes	A metodologia XP credita uma grande parte do seu sucesso na escrita dos testes de unidade e testes funcionais. Os testes de unidade servem para dar ao programador a certeza de que o que ele está programando está correto. Os testes funcionais servem para comprovar junto ao cliente que as funcionalidades estão sendo implementadas da forma como o cliente quer e livre de erros. Em um projeto para a Web, sabe-se a importância dos requistos não funcionais. Estes requisitos constituem restrições do ambiente onde o sistema irá operar (segurança, tráfego, tipo de cliente, concorrência, sistemas de apoio) e podem ser fundamentais para o bom funcionamento e sucesso da aplicação. Logo, o teste desses requisitos deve ser feito colocando-se o sistema para operar num ambiente muito parecido com o ambiente real onde ele será usado. Portanto, quando criada uma adaptação de XP para a Web Engineering, esta deve considerar os testes de requisitos não funcionais. Existem diversas ferramentas disponíveis no mercado que possibilitam a automação de diversos tipos de testes como: testes unitários [61], testes funcionais [62], e testes Web [63].
Distribuição "Deployment"	As questões referentes à distribuição dos componentes que fazem parte do sistema Web (módulos executáveis, páginas HTML, Bancos de dados, etc...) devem ser cuidadosamente tratadas. Pela leitura de [28,29] nota-se que esse assunto não é muito comentado e fica, portanto necessário se estabelecer recomendações e "guidelines" que sirvam para guiar as organizações nessa tarefa.
Manutenção e evolução	Da mesma forma que a disciplina anterior, essa disciplina de suma importância precisará ser tratada numa adaptação de XP para Web através de um conjunto de "guidelines" desenvolvidas para guiar o processo de manutenção das aplicações Web. Deve-se ter cuidado com a forma de alteração dos componentes que formam o sistema Web que podem ser de diversos tipos (HTML, código fonte, etc.).

A Tabela 5.1 acima e tudo que foi discutido nos capítulos 2 e 3 servem como base para a elaboração do processo XWebProcess a ser feita nas seções 5.3 e 5.4.

5.2.4 Extensões de XP

A possibilidade de extensão e alteração de Extreme Programming é um assunto polêmico que possui duas correntes de opinião conflitantes. Segundo seus próprios autores e criadores, XP não pode ser alterado, pois eles afirmam que se qualquer nova prática for inserida ou uma prática existente de XP não for utilizada a equipe não estará usando XP [28,29,30,33]. No entanto, outros autores discordam desta opinião e acreditam que XP possa ser estendido e adaptado para certas situações e mesmo assim seus valores e princípios seriam respeitados [60,64].

Para não entrar em conflito com nenhuma destas duas correntes, pode-se adotar uma solução já sugerida por Ambler em [34]. A idéia é então utilizar XP como base para o processo de engenharia de software para Web que deverá preservar as características ágeis de XP respeitando seus princípios e valores. Além disto, tal processo possuirá nova estrutura que levará em consideração os problemas de sistemas Web e que resultará na incorporação ou adaptação de novas práticas e papéis. Logo, tal processo não contraria nenhuma das correntes anteriores, pois ele não é XP modificado, mas sim um novo processo chamado XWebProcess que será mostrado nas próximas seções.

5.3 Concepção de XWebProcess

O processo ágil de desenvolvimento de aplicações Web, XWebProcess, foi concebido tendo por base teórica os levantamentos feitos ao longo deste e dos dois capítulos anteriores. Os dois principais componentes de XWebProcess (mostrados na Figura 5.1) são justamente:

1. O processo ágil Extreme Programming (Capítulo 3), no qual XWebProcess irá se basear e criar as adaptações necessárias (novos papéis, disciplinas, atividades, etc.) para o desenvolvimento Web;

2. O estudo da engenharia de software para a Web (Capítulo 2) e da contextualização de XP (Seção 5.2) que serve para levantar as necessidades que o processo XWebProcess deve tratar.

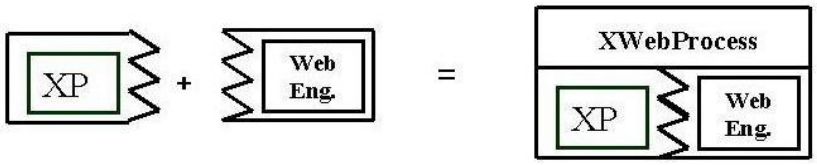


Figura 5.1 - Concepção de XWebProcess

A concepção de XWebProcess é detalhada nas seções seguintes que explicam o processo através da sua descrição em modelos feitos em SPEM [48]. A utilização de modelagem conforme já explicado no capítulo anterior se justifica pela necessidade de uma melhor forma para descrever o processo facilitando o seu entendimento e a sua adaptação.

5.4 XWebProcess

A criação de XWebProcess tomou por base a modelagem de XP descrita e apresentada nas seções 4.3 e 4.4. Surgiram novos elementos estruturais como papéis, atividades e artefatos, além de que alguns elementos existentes foram modificados para atender as necessidades do desenvolvimento para a Web. Além disso, as necessidades levantadas na Seção 5.2 serviram para guiar a criação de XWebProcess.

É importante mencionar que XWebProcess será descrito neste capítulo através de duas perspectivas: dinâmica e estrutural. A perspectiva dinâmica (Seção 5.4.1) mostra um diagrama de atividades semelhante ao da Figura 4.4 dando destaque ao que foi adicionado ou modificado na modelagem feita em XP no capítulo anterior. A perspectiva estrutural (Seção 5.4.2) mostra o detalhamento, através de diagramas de classes estereotipados, de todas as disciplinas inseridas ou modificadas em XWebProcess. As disciplinas não modificadas não serão descritas, pois já o foram no capítulo anterior.

5.4.1 Modelagem Dinâmica de XWebProcess

Com base no que foi discutido anteriormente sobre o que um processo de referência de desenvolvimento de aplicações Web deve conter (Seção 2.7) e sobre a contextualização de XP neste sentido (Seção 5.2) é que foram feitas as modificações que deram origem ao processo XWebProcess.

A Figura 5.2 apresenta o processo XWebProcess sob um ponto de vista dinâmico. Cada elemento gráfico mostrado na figura corresponde a um elemento disciplina de SPEM. O detalhamento de cada componente é explicado na seção seguinte e é importante explicar o que muda em XP de um ponto de vista "top-down". A Tabela 5.2 explica quais disciplinas são incluídas e modificadas (aparecem em destaque na Figura 5.2) em relação as disciplinas de XP mostradas no capítulo anterior (Figura 4.4).

Tabela 5.2 - Disciplinas inseridas e modificadas na modelagem de XP

DISCIPLINA	TIPO DE EXTENSÃO	DESCRIÇÃO E JUSTIFICATIVA
Fazer Explorações Iniciais	Modificada	Durante as explorações iniciais são feitos testes com possíveis arquiteturas e tecnologias de implementação Web procurando investigar sua viabilidade. Além disso, pode-se usar técnicas como prototipação para esclarecer dúvidas sobre requisitos iniciais junto ao cliente.
Definir e Revisar Requisitos	Modificada	Esta modificação procura estabelecer uma arquitetura inicial que servirá de base para a construção do sistema Web. A metáfora pode ser utilizada, mas como um complemento a arquitetura que tem uma importância muito grande em sistemas Web.
Fazer Projeto	Modificada	Esta modificação introduz uma atividade de projetar camada de dados tão importante em aplicações Web. XP não deixa explícito o ponto onde isso é feito e por isso essa disciplina foi alterada.
Projetar navegação e interface gráfica Web	Inserida	Esta extensão foi introduzida devido à relevância que a navegação e a interface gráfica possuem no desenvolvimento Web. As interfaces gráficas Web podem ser bastante complexas incluindo gráficos, vídeos, animações, etc. Além disso, elas podem oferecer diversos caminhos distintos para navegar pelo seu conteúdo.
Testes Web	Inserida	Esta extensão se refere aos testes feitos para simular o ambiente operacional onde a aplicação Web é usada. Devem ser levados em consideração diversos requisitos não funcionais como carga da rede, performance, número de usuários, etc.
Manutenção Web	Inserida	Esta extensão se refere principalmente a manutenção do conteúdo do Web Site que pode mudar constantemente e a manutenção dos arquivos que compõe o site.

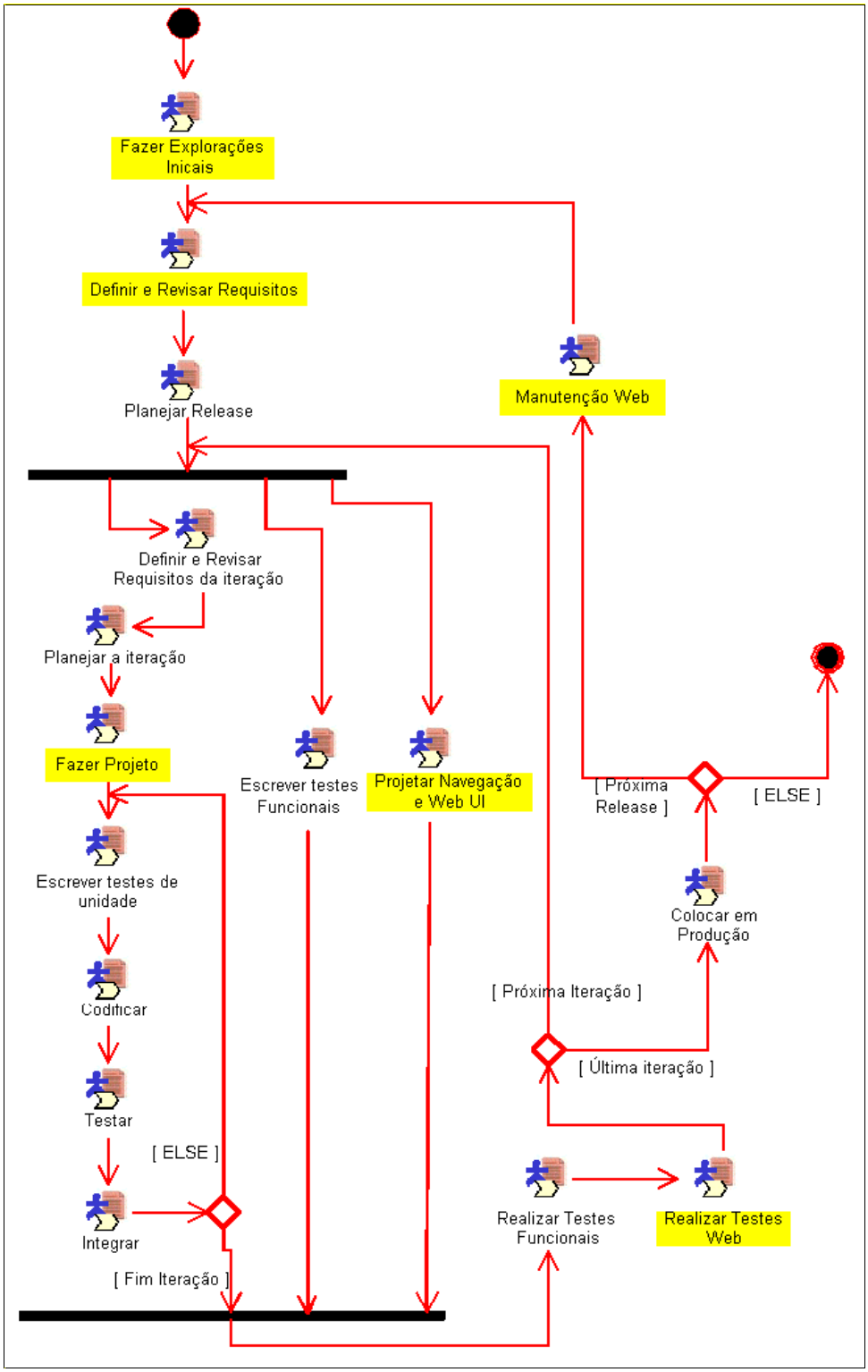


Figura 5.2 – XWebProcess sob ponto de vista dinâmico.

A Figura 5.2 mostra quando as novas disciplinas são executadas durante um projeto que siga o processo XWebProcess. O que é importante detalhar agora é o que muda dinamicamente devido às disciplinas introduzidas que são:

- **Projetar navegação e interface gráfica Web:** Esta disciplina será executada paralelamente à atividade de escrever testes funcionais;
- **Realizar Testes Web:** Esta disciplina é executada após a realização dos testes funcionais para verificar interoperabilidade de browsers, carga na rede, etc;
- **Manutenção Web:** Esta disciplina começa a ser executada a partir da entrega do primeiro release do sistema ao cliente.

O detalhamento sobre cada disciplina é mostrado na próxima seção que descreve os papéis envolvidos, as atividades realizadas e tudo aquilo que envolve os elementos de cada disciplina.

5.4.2 Modelagem Estática de XWebProcess

Esta seção descreve o detalhamento da estrutura de cada elemento inserido e modificado na modelagem de XP conforme apresentado na seção anterior. Para isso, são utilizados estereótipos de SPEM na construção dos diagramas de classes. Cada diagrama detalha uma disciplina e respeita as restrições de SPEM. Antes de mostrar cada diagrama é importante descrever os novos papéis (Roles) que surgem em XWebProcess mostrados na Tabela 5.3.

Tabela 5.3 - Novos papéis de XWebProcess

PAPEL (ROLE)	DESCRIÇÃO
Arquiteto	Constrói a arquitetura, ou seja, a estrutura sobre a qual a aplicação Web será construída. Possui conhecimentos de análise e projeto, modelos de arquiteturas (camadas, cliente-servidor, etc.) e de tecnologias de desenvolvimento para a Web. Também será responsável pelo projeto da camada de dados.
Web Designer	Possui conhecimento de construção de páginas Web e de programação. Será responsável pela geração do protótipo do site e do projeto da sua navegação e interface gráfica.

PAPEL (ROLE)	DESCRIÇÃO
Administrador do site	Fica responsável pela distribuição, manutenção e configuração dos arquivos (páginas HTML, programas, bancos de dados, arquivos de configuração) nos servidores correspondentes. Ou seja, se responsabiliza pela a infra-estrutura necessária no ambiente de operação da aplicação Web.

A descrição dos papéis é importante, pois eles aparecem nos diagramas apresentados a seguir. Conforme dito anteriormente, cada diagrama de classe representa uma disciplina do diagrama de atividades mostrado na Figura 5.2. As figuras mostradas abaixo mostram o diagrama e a explicação sobre os elementos modificados e inseridos descritos anteriormente na Tabela 5.2.

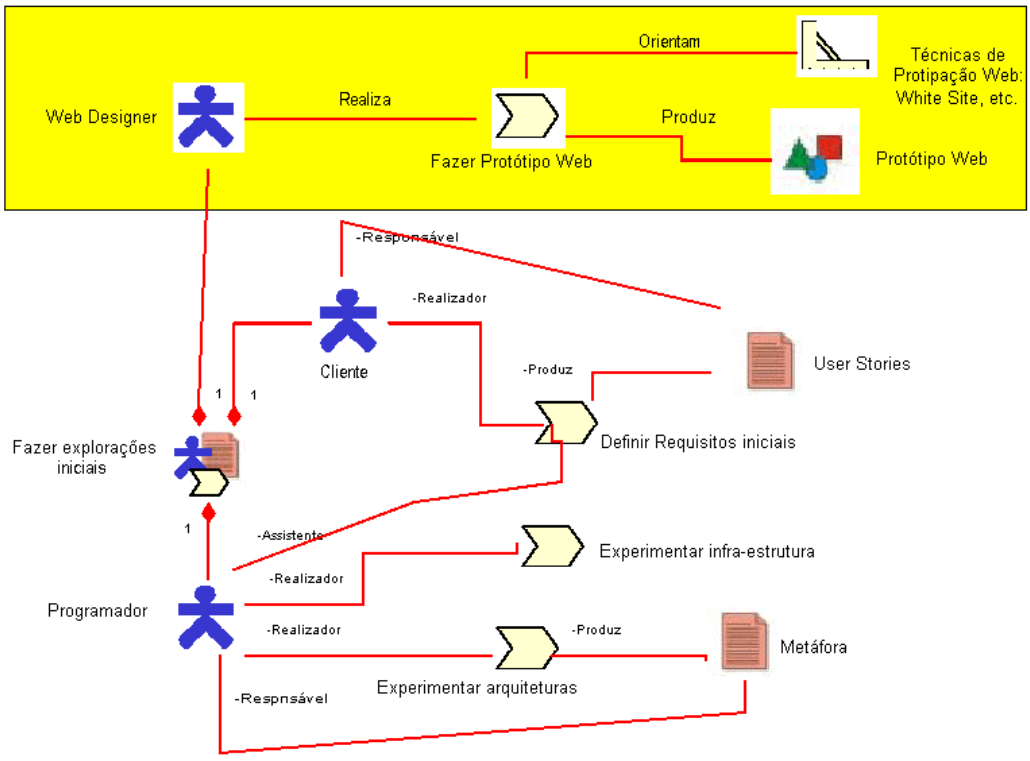


Figura 5.3 – Disciplina modificada: Fazer Explorações Iniciais

Fazer Explorações Iniciais: A extensão nesse caso foi feita através de uma modificação representada pela parte em destaque no diagrama. O Web designer executa a atividade fazer protótipo Web com a orientação de técnicas de prototipação de Web Sites como, por exemplo, a técnica “White Site Prototyping”. A realização dessa

atividade produz um protótipo Web como resultado. A prototipação em sistemas Web é apontada em [4,39] como um importante meio de esclarecer dúvidas sobre os requisitos e sobre o negócio em si que o sistema vai representar. Estas dúvidas são muito comuns em estágios iniciais de projetos Web onde não se sabe efetivamente quais os principais objetivos do sistema. É importante destacar que mesmo que não seja possível envolver os diversos tipos de cliente que possam vir a existir, é importante que se tenha alguém na equipe que possa representar este papel, ou seja, algum especialista no tipo de negócio em questão.

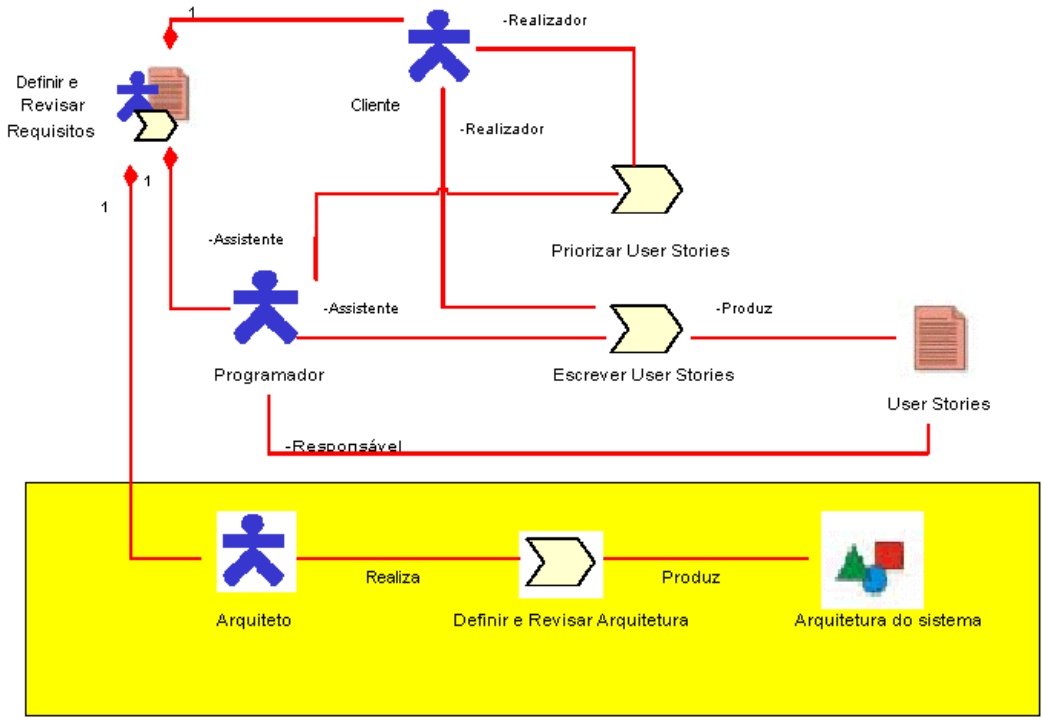


Figura 5.4 - Disciplina modificada: Definir e Revisar Requisitos

Definir e Revisar Requisitos: A extensão nesse caso foi feita através de uma modificação representada pela parte em destaque no diagrama. O arquiteto é responsável pela execução da atividade de Definir e Revisar arquitetura e ao realizar tal atividade produz como resultado um modelo que representa a arquitetura do sistema. O objetivo é projetar uma arquitetura que possa ser flexível o bastante para permitir mudanças futuras e que de preferência se baseie em padrões arquiteturais em camadas

como, por exemplo, os padrões vistos em [41,43]. Alguns trabalhos [4,39] apontam que a arquitetura desempenha um papel muito importante em sistemas Web. Isso se deve ao fato que é comum que os sistemas Web sejam constantemente integrados com outros sistemas (parceiros, clientes, fornecedores) que podem ser implementados usando diversas tecnologias distintas o que requer flexibilidade para incorporar novas tecnologias e serviços e facilitar a manutenção dos sistemas existentes.

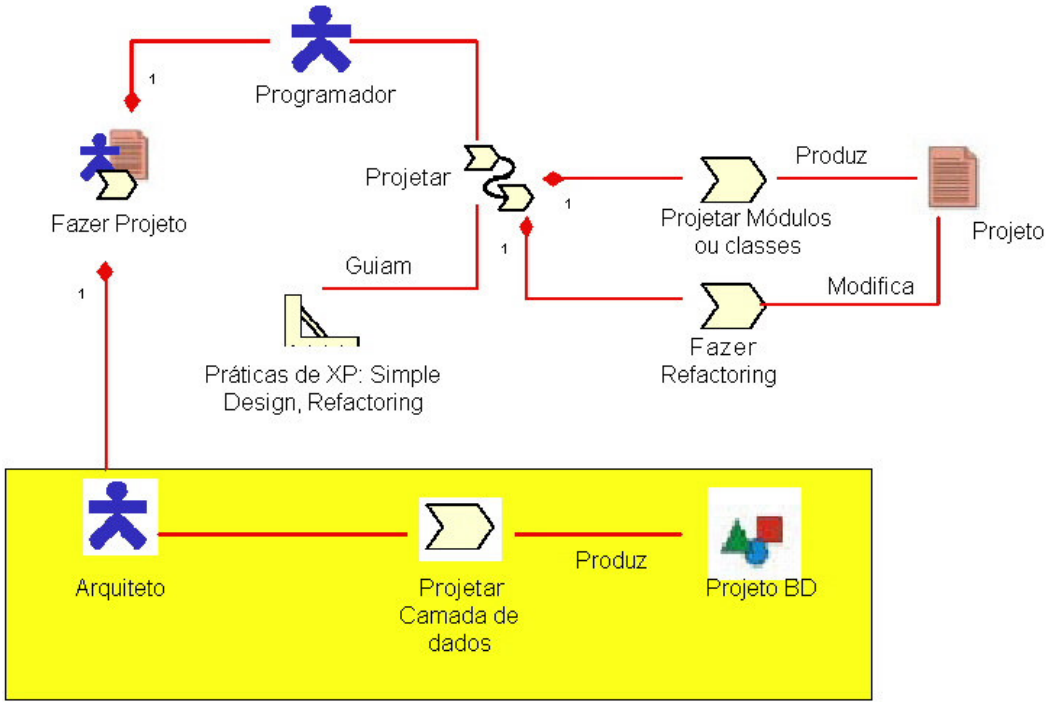


Figura 5.5 - Disciplina modificada: Fazer Projeto

Fazer Projeto: Essa disciplina foi alterada para incluir a atividade de projetar a camada de dados realizada pelo arquiteto. O projeto e criação do esquema do banco de dados são independentes de tecnologia ou paradigma, ou seja, pode se escolher um BD relacional ou orientado a objeto. A inclusão desta atividade se justifica pela necessidade que a maioria dos sistemas Web têm de armazenar grandes quantidades de informação. Isso não quer dizer que XP não trate isso, porém pela sua leitura não fica explícita esta necessidade, o que nos levou a criar a modificação na disciplina.

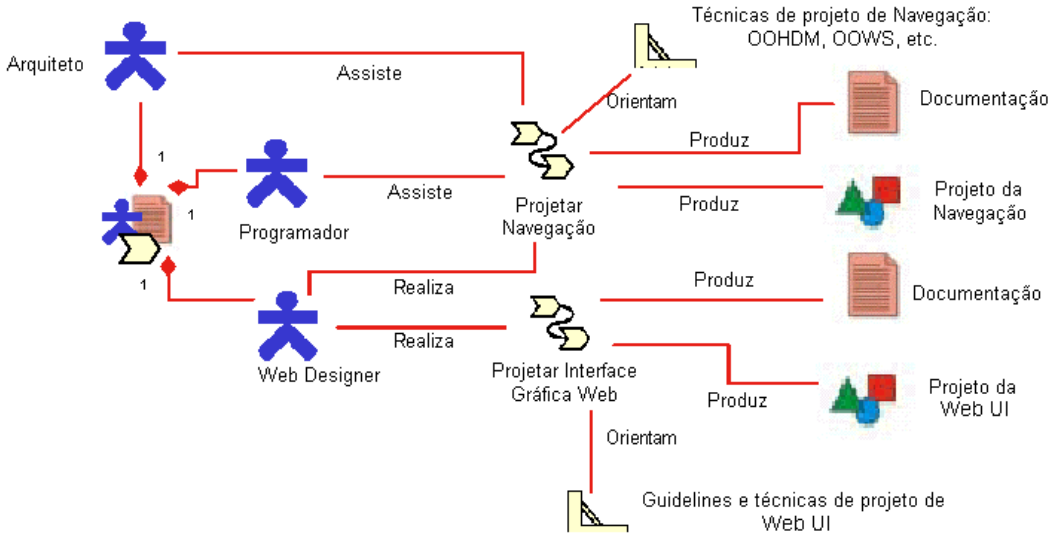


Figura 5.6 - Disciplina Incluída: Projetar navegação e interface gráfica Web

Projetar navegação e interface gráfica Web: Esse componente inclui três papéis: programador, arquiteto e Web designer sendo, os últimos dois, papéis novos. O Web designer é responsável por realizar as atividades referentes a Projetar Navegação com a assistência do programador e do arquiteto. Tais atividades são de grande importância para aplicações Web, pois têm o objetivo de projetar a forma como se pode navegar para obter informações e interagir com o sistema Web. Existem diversas técnicas como OOHDM, OOWS e técnicas informais, apresentadas na Seção 2.4, que podem dar uma orientação de como projetar a navegação do sistema Web.

Outro conjunto de atividades importante é o de Projetar a Interface Gráfica Web. O responsável por tais atividades também é o Web designer. As atividades aqui procuram criar tudo o que se refere à interface gráfica do Web Site como: páginas HTML, animações, figuras, imagens, gráficos, sons e vídeo, ou seja, todos os recursos e mídias que fazem parte do Web Site. Para auxiliar tais atividades as pessoas podem seguir algumas técnicas e guias conforme pode ser visto em [70]. A inclusão desta disciplina se torna fundamental, pois a importância da interface gráfica em aplicações Web é muito grande visto que são bastante complexas (incluem gráficos, sons, vídeos), além do que um site desorganizado e difícil de navegar simplesmente deixa de ser acessado pelos usuários.

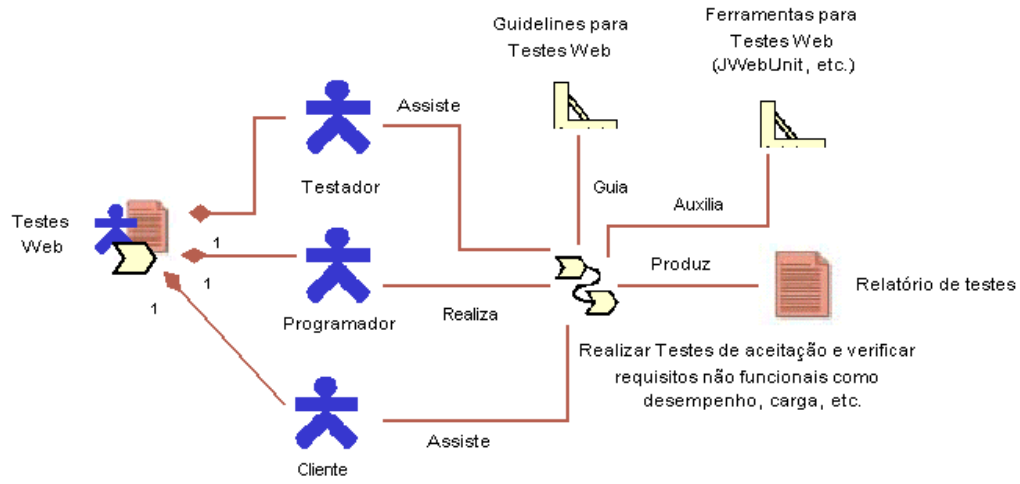


Figura 5.7 - Disciplina incluída: Testes Web

Testes Web: Esta disciplina possui três papéis (administrador do site, programador e cliente) sendo o primeiro um papel novo. O programador realiza as atividades referentes aos testes Web com a assistência do cliente que irá verificar se o sistema atende as suas exigências e do administrador do site que irá ajudar a configurar o ambiente operacional dos testes. Os testes aqui devem tentar simular situações que podem ocorrer no ambiente real onde o sistema irá operar, de forma a verificar se o sistema Web atende a requisitos não funcionais como desempenho, balanceamento de carga e quantidade de acessos. Ou seja, deve-se verificar se o sistema Web irá operar de forma adequada no seu ambiente operacional. Esta tarefa pode ser auxiliada por ferramentas de teste Web como JWebUnit [63].

Os testes em alguns tipos de sistemas Web podem ser bem mais complexos do que em outros tipos de sistemas, pois alguns requisitos não funcionais relativos ao desempenho do sistema devem ser observados. É muito comum ver casos práticos, como por exemplo, sites de serviço público (Receita Federal, INSS, etc.) onde perto do prazo de encerramento do serviço o sistema fica sobrecarregado e não consegue atender a demanda dos usuários. Portanto, esta disciplina também é muito importante para o caso de desenvolvimento Web.

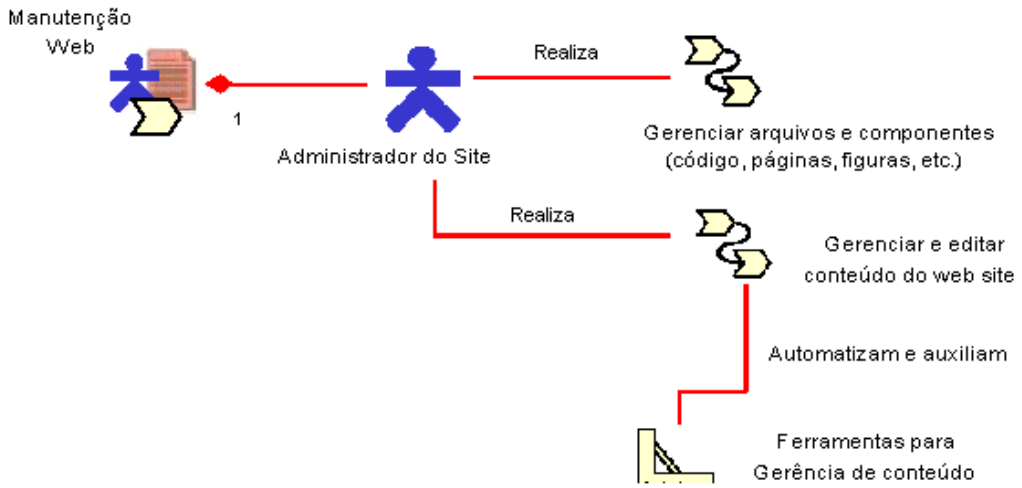


Figura 5.8 - Disciplina incluída: Manutenção Web

Manutenção Web: Esta disciplina possui um papel novo: administrador do site, que se preocupa com a editoração e gerência do conteúdo do Web site. Isso é extremamente importante, pois em alguns tipos de sistemas Web o conteúdo pode variar freqüentemente como, por exemplo, em sites de portais, revistas e jornais. Ele pode se beneficiar do uso de ferramentas de gerência de conteúdo para facilitar o seu trabalho. Além disso, ele realiza atividades para dar manutenção e gerenciar os componentes que fazem parte da infra-estrutura de hardware e software do sistema Web como sistemas operacionais, SGBDs, middlewares, servidores Web, arquivos componentes do sistema (páginas HTML, servlets, ASPs, etc.). Ou seja, ele deve se preocupar em como distribuir os componentes fisicamente nos servidores, como configurá-los e como atualizá-los em caso de modificações. Uma boa organização dos componentes de um Web site ajuda na sua evolução e gerência do seu conteúdo. Deve-se pensar em seguir uma política bem definida para a localização e nomenclatura dos componentes nos diversos servidores.

Com base na modelagem dinâmica e estática apresentada nesta seção e na anterior, pode-se concluir que o processo XWebProcess pode ser adotado em uma organização que queira usar um processo ágil de desenvolvimento, e que pretenda aplicar o processo na construção de sistemas para a Web. O processo de instanciação, ou seja, uso real do processo deve tomar alguns cuidados para não tornar o processo

muito pesado o que contraria a filosofia de XP. Algumas dicas e guias sobre como fazer a instanciação de XWebProcess são explicadas na próxima seção.

5.5 Aplicando XWebProcess em Projetos

O processo XWebProcess, conforme descrito na seção anterior, foi concebido com base em Extreme Programming e no estudo da engenharia de software de aplicações Web. Agora, é importante explicar em quais tipos de aplicações e condições XWebProcess pode ser aplicado (Seção 4.6.1). Além disso, foi mencionado anteriormente que XWebProcess também pode ser adaptado para atender a novas características não previstas na sua concepção. Para fazer tais adaptações devem ser respeitadas algumas regras que foram explicadas na Seção 4.6.2.

5.5.1 Tipos de aplicações mais adequadas

XWebProcess é um processo de desenvolvimento de aplicações Web que procura ser o mais genérico possível. XWebProcess procura ser independente de qualquer tecnologia, técnica ou método específico. Devido a essa característica XWebProcess pode não ser adequado para o desenvolvimento de certos tipos de aplicações Web. Nestes casos o processo pode ser adaptado para que sejam criados novos elementos que tratem as características específicas conforme será explicado na próxima seção.

Alguns dos tipos de aplicações que não são adequadas para XWebProcess são: **Aplicações Web de tempo real:** aplicações de Telemedicina, vídeo sob demanda, vídeo conferencia, WebTV, etc; e **Engenhos de Busca.** Estes tipos de aplicações exigem tratamentos e técnicas especiais que não fazem parte dos objetivos iniciais de concepção de XWebProcess.

Os tipos de aplicações mais adequados para serem construídos por XWebProcess são aplicações de comércio eletrônico (B2B, B2C) assim como

aplicações Web de outros domínios que não necessitem de tratamentos especiais como as citadas anteriormente. Exemplos de aplicações adequadas podem ser: Venda de produtos pela Web, Atendimento ao cliente pela Web, prestação de serviços públicos pela Web e diversas outras aplicações.

Portanto, XWebProcess é um processo genérico de desenvolvimento de aplicações Web que pode ou não ser adequado para determinado tipo de aplicação Web. A apresentação de alguns exemplos nesta seção serve como auxílio para identificar se uma dada aplicação é ou não adequada. Caso não seja, o processo XWebProcess pode ser adaptado conforme será explicado na próxima seção.

5.5.2 Adaptação e instanciação de XWebProcess em Projetos

Esta seção trata de uma importante parte que é esclarecer como XWebProcess pode ser aplicado em projetos de desenvolvimento de sistemas Web na prática. Conforme mencionado na seção anterior XWebProcess procura ser o mais genérico possível na sua definição e por causa disso, em algumas ocasiões ele precisa ser adaptado para atender a requisitos específicos do projeto.

É importante esclarecer agora o que vem a ser a adaptação e a instanciação de XWebProcess. A definição de **adaptação** nesta dissertação se refere às modificações necessárias nos elementos do processo para atender a um requisito em particular. Exemplos de adaptações já foram mostrados anteriormente através da extensão de XP, criando novos elementos, como papéis e disciplinas que vieram a gerar XWebProcess.

A **instanciação** pode ser considerada uma aplicação particular do processo em um determinado projeto. Por exemplo, o processo XWebProcess define na disciplina de projetar navegação e interface gráfica Web (Figura 5.6) que uma técnica como OOWS, OOHDM, ou uma técnica informal de projeto Web pode ser aplicada. O processo, no entanto, não informa qual deve ser a técnica escolhida e nem como ela deve ser aplicada. Esta decisão vai ficar a cargo da equipe que faz parte do projeto que vai escolher a técnica de projeto que considere mais adequada as suas habilidades e assim

instanciar o processo XWebProcess. Portanto, a instanciação é o ato de tornar concreto o processo em um projeto específico.

Tanto a adaptação quanto a extensão de XWebProcess não podem ser feitas de qualquer maneira, pois devem preservar a característica ágil do processo. Diante disso são elaboradas algumas guias que procuram esclarecer o que deve ser considerado ao fazer adaptações e extensões em XWebProcess. Estas guias são mostradas abaixo.

- **Agilidade[9] na adaptação de XWebProcess:** Ao se fazer uma adaptação em XWebProcess deve se considerar quais elementos do processo deverão sofrer modificações e quais novos elementos serão inseridos. Toda adaptação de XWebProcess deve ser feita visando a preservação das características ágeis do processo.
- **Agilidade na instanciação de XWebProcess:** Ao instanciar XWebProcess para um projeto em particular deve-se levar em consideração se as técnicas, métodos e ferramentas escolhidas possibilitam um trabalho eficiente e ágil para não se desviar do foco principal de um processo ágil que é o de gerar código e não documentação excessiva.
- **Independência na adaptação:** A adaptação de XWebProcess não pode tornar o processo dependente de uma tecnologia, método ou ferramenta em particular.

As guias definidas servem para esclarecer o que uma adaptação ou uma instanciação do processo XWebProcess devem respeitar. Um exemplo concreto de aplicação de XWebProcess para o desenvolvimento de uma aplicação Web será mostrado no próximo capítulo.

[9] A definição de agilidade considerada aqui fica a critério da equipe que for utilizar XWebProcess. Se uma determinada equipe X, por exemplo, sabe utilizar a técnica OOWS de forma eficiente para projetar a navegação da aplicação Web e sabe que a técnica pode trazer benefícios para o projeto como a geração de parte do código Web com o uso de ferramentas específicas, então tal técnica neste caso não tornou o processo lento e nem desviou o foco de XWebProcess.

5.6 Considerações Finais

Este capítulo apresentou o processo XWebProcess. Este processo, que é um processo ágil de desenvolvimento de aplicações Web, foi elaborado com base em Extreme Programming e em um levantamento das principais características de aplicações Web que um processo deve levar em consideração.

XWebProcess, apresentado anteriormente sob as perspectivas dinâmica e estrutural, se baseia na modelagem produzida nas seções 4.3 e 4.4. A modelagem de XWebProcess com SPEM trouxe benefícios, como por exemplo, a organização dos elementos do processo (papéis, artefatos, etc.) em modelos que permitem facilitar o entendimento sobre o funcionamento e a estrutura do processo sob perspectivas distintas (dinâmica e estática).

Para facilitar a criação do processo foi importante realizar a modelagem de Extreme Programming na linguagem SPEM o que tornou mais fácil a compreensão do relacionamento dos elementos de XP (artefatos, atividades, disciplinas, papéis) tanto de um ponto de vista dinâmico, através da modelagem de XP com diagrama de atividade e estereótipos SPEM, quanto também de um ponto de vista estático, através dos diagramas de classe estereotipados. Com base na modelagem feita foram modificados e inseridos novos elementos em XP (disciplinas) gerando o processo XWebProcess mostrado também sob um ponto de vista “top down” através do diagrama de atividades e detalhado através dos diagramas de classes.

Além disso, o fato de que o atual estado da arte da engenharia de software para Web necessita de processos de software adequados e eficientes para a construção de aplicações Web, faz com que o processo aqui apresentado possa atender a essa demanda. Depois da elaboração e descrição de XWebProcess foi comentado acerca de como este processo pode ser adaptado e instanciado em projetos e o que deve ser considerado para tal. Um exemplo de aplicação será mostrado no próximo capítulo onde XWebProcess é aplicado em um projeto real de desenvolvimento de uma aplicação Web através de um estudo de caso.

6
Análise Experimental

Este capítulo apresenta um experimento realizado com XWebProcess. A Seção 6.1 mostra uma visão geral sobre experimentos em engenharia de software. A Seção 6.2 descreve o experimento conduzido para validação desta dissertação. A Seção 6.3 mostra a análise e interpretação dos resultados obtidos no experimento, enquanto a Seção 6.4 apresenta as considerações finais do capítulo.

6.1 Visão geral

Ao definir um processo de desenvolvimento de software é importante investigar sua utilização na prática para avaliar sua utilidade e seus benefícios. Em [75-82] são descritas técnicas de validação de processos, métodos e ferramentas de engenharia de software. Dentre estas técnicas, as mais utilizadas são experimentos, estudos de caso, questionários e entrevistas.

Questionários e entrevistas são boas técnicas para o levantamento de dados qualitativos e para saber a opinião de pessoas envolvidas em um determinado projeto. Por outro lado, experimentos e estudos de caso são mais utilizados no levantamento de dados quantitativos para fazer determinadas avaliações, comparações e análises estatísticas sobre uma determinada situação prática. A diferença entre experimento e estudo de caso é um tanto quanto sutil e, segundo [75,80], a principal diferença entre estas duas técnicas está no grau de controle da situação a qual se quer investigar.

Em um experimento existe um grau de controle maior e pode-se controlar as variáveis envolvidas no problema. Por exemplo, se o objetivo for investigar se o método de projeto A é melhor do que o B, existe a possibilidade de controlar quais grupos utilizam A ou B, quais ferramentas CASE, ambientes de desenvolvimento e linguagens

de programação os grupos usam. Além disso, existe a possibilidade de fazer interferências controladas no experimento, como fazer alterações nos requisitos do sistema e retirar membros da equipe.

Por outro lado, em um estudo de caso o controle é menor. Não se tem controle sobre as variáveis envolvidas no problema e não existe uma interferência direta sobre a situação avaliada. Geralmente estudos de caso são conduzidos através de um acompanhamento daquilo que está ocorrendo na prática, no sentido de fazer uma avaliação ou comparação, mas sem a liberdade existente no experimento para interferir e controlar a situação.

Portanto, estudos de caso podem ser mais adequados para a avaliação de projetos reais em empresas, enquanto experimentos são mais adequados para o que se chama de "situações de laboratório", em ambientes acadêmicos ou não, onde simulações são feitas com o objetivo de avaliar um processo ou técnica em particular. Devido ao seu grau de controle maior, os experimentos são uma técnica mais rica de avaliação e possibilitam uma melhor interpretação e generalização de seus resultados [80].

Fazer experimentos e estudos de caso na área de engenharia de software não é uma tarefa fácil devido à existência de diversos fatores que podem influenciar negativamente a realização dos mesmos de forma que não se possa tirar conclusões satisfatórias.

Um exemplo que pode ser mostrado para esclarecer isso é o seguinte: suponha que se queira comparar dois métodos de projeto A e B utilizando-se dois grupos de pessoas que irão resolver um determinado problema, para saber qual método produz software de melhor qualidade. Existem alguns fatores que podem influenciar esse experimento, tais como:

- Experiência das pessoas com os métodos A e B;
- Ferramentas utilizadas pelos dois grupos;
- Tipo do problema a ser resolvido. Pode ser que um ou outro método seja mais apropriado para um determinado tipo de problema.

No entanto, apesar do problema de conduzir experimentos e estudos de caso em engenharia de software ser complexo, existem maneiras adequadas de fazer isso conforme explicado em [75-82].

Diante disso, este capítulo mostra como foi feita a validação de XWebProcess. Um experimento foi conduzido para avaliar a agilidade e eficiência de XWebProcess. O objetivo deste experimento é verificar quantitativamente que o processo continua ágil apesar de inserir e modificar elementos em XP. Além disso, uma análise qualitativa do processo foi realizada através da aplicação de questionários com as pessoas envolvidas no experimento, objetivando investigar os benefícios advindos das adaptações realizadas no processo. As seções seguintes descrevem o experimento conduzido e as análises realizadas.

6.2 Descrição do experimento

Esta seção mostra a avaliação feita de dois processos ágeis XP e XWebProcess, no contexto de um estudo empírico realizado para comparar esforço nos dois processos. Um experimento foi planejado e conduzido para demonstrar que XWebProcess, apesar de adicionar e adaptar elementos de XP, é um processo ágil para o desenvolvimento de aplicações Web.

O experimento foi planejado antes da sua realização de acordo com o que é definido em [75-82]. Durante o experimento, foram coletados dados sobre o esforço gasto em cada disciplina e artefato de ambos os processos (XP e XWebProcess) para comparar as suas principais diferenças. A Seção 6.2.1 descreve em detalhes como o experimento foi organizado e a Seção 6.2.2 mostra como os dados de esforço foram coletados.

6.2.1 Organização do experimento

O projeto utilizado na realização do experimento consistiu da implementação de uma aplicação Web de porte médio. Esta mesma aplicação foi implementada por oito grupos de alunos organizados em equipes de cinco pessoas. Todas as pessoas envolvidas no experimento eram alunos do oitavo semestre do curso Ciência da Computação da UFPE e o projeto se tornou parte da nota da disciplina Engenharia de Software. O sistema implementado consistiu de uma livraria virtual com os seguintes requisitos funcionais:

- **[RF01] Cadastro de empregados**. Consiste das operações de inserção, remoção e atualização dos empregados da livraria. Somente um empregado válido e logado (login e senha válidos) no sistema pode operar esta funcionalidade;
- **[RF02] Cadastro de livros**. Consiste das operações de inserção, remoção e atualização dos livros da livraria. Novamente, somente um empregado válido e logado no sistema pode operar esta funcionalidade;
- **[RF03] Cadastro de clientes**. Consiste das operações de inserir e atualizar os dados dos clientes. O cliente é responsável por registrar a si mesmo e somente após este cadastro pode realizar suas compras na livraria virtual através do seu login e senha;
- **[RF04] Venda de livros.** Um cliente já registrado e logado no sistema pode efetuar normalmente suas compras na livraria. O processo de compra consiste de adicionar livros a uma cesta de compras até que o cliente decida encerrar a compra. Isso é confirmado pela geração de uma resposta de confirmação juntamente com um código da compra que pode ser usado para consulta pelo cliente.

Algumas restrições importantes do experimento foram:

- O experimento envolveu quarenta pessoas divididas em oito grupos de cinco pessoas. Quatro grupos implementaram o sistema usando XP e os outros quatro

usando XWebProcess. A escolha de qual grupo iria usar qual processo foi randômica;

- O experimento foi conduzido de forma independente por cada grupo. Não existiu comunicação entre membros de grupos diferentes e cada grupo teve a liberdade de escolher seu local e horário de trabalho. Apesar desta liberdade cada membro da equipe tinha um papel bem definido a seguir dentro do processo e o esforço gasto nas atividades eram catalogados em uma planilha de esforço descrita adiante;
- As tecnologias escolhidas para a implementação foram: JSP e Servlets para a camada de apresentação, Java para a camada de negócios e JDBC para a camada de dados. Todos os grupos usaram estas mesmas tecnologias;
- A experiência dos membros dos grupos era de certa forma homogênea uma vez que eram alunos da mesma universidade e semestre além de terem "background" e experiência de trabalho semelhantes;
- Todos os alunos tinham experiência prévia com Java (mais de dois anos) e banco de dados (mais de um ano);
- Poucos alunos tinham experiência prévia com as tecnologias Web escolhidas (JSP e Servlets) e acabaram aprendendo durante o experimento;
- Automação de testes com JUnit foi explicado antes do experimento começar e foi usado por todos os grupos;
- O tempo de entrega do sistema foi de vinte dias, divididos em duas iterações de dez dias.

Os grupos foram apresentados a cada processo assistindo a aulas antes da realização do experimento. Os capítulos 3, 4 e 5 e o Apêndice B deste trabalho foram disponibilizados como material de leitura para os seus respectivos grupos. Uma observação importante é que os alunos que usaram o processo XP não assistiram as aulas de XWebProcess antes do experimento, o que poderia influenciá-los.

Todos os alunos eram responsáveis por um ou mais papéis, tais como programador, testador, Web designer, rastreador, cliente e arquiteto. O autor desta dissertação participou como cliente, ajudando na definição dos requisitos do sistema e também como gerente de projeto para verificar se os processos estavam sendo seguidos corretamente.

As duas iterações foram organizadas de forma semelhante e sempre contemplavam três reuniões envolvendo o autor desta dissertação e cada uma das equipes. No início de cada iteração era realizada uma reunião para a definição dos requisitos da iteração. O autor atuava no papel de cliente e ajudava na definição das estórias a serem implementadas. Posteriormente, no meio da iteração, era marcada uma reunião de acompanhamento para verificação do andamento do projeto. No final da iteração era realizada uma última reunião para validação das estórias implementadas junto ao cliente.

O principal objetivo do experimento era comparar XP e XWebProcess em relação a esforço. A intenção era provar que as adaptações feitas para a criação de XWebProcess não representam um impacto significativo na agilidade do processo, o que será mostrado pela análise quantitativa descrita posteriormente. Além disso, a análise qualitativa que foi realizada através de questionários após o término do projeto procura destacar que as adaptações feitas em XWebProcess realmente trazem benefícios e que este é um processo mais adequado que XP ao desenvolvimento Web. A definição do experimento pode ser resumida no "goal definition template" abaixo [80,81].

Objeto de estudo: XP e XWebProcess.

Propósito: Avaliar o uso de ambos os processos no desenvolvimento de aplicações Web.

Foco de Qualidade: Esforço

Perspectiva: Desenvolvedores de sistema.

Contexto: O experimento foi realizado por quarenta estudantes de graduação da UFPE, como sujeitos, divididos em oito grupos de cinco pessoas cada. Quatro grupos utilizaram XWebProcess e os outros quatro XP para implementar o mesmo sistema de

livraria Web. Todos os grupos tiveram treinamento prévio em JUnit e nos seus respectivos processos. Os grupos de XP só tiveram treinamento em XWebProcess após o fim do experimento com a finalidade de responder o questionário.

6.2.2 Como os dados foram coletados

Durante a realização do projeto, cada grupo coletou dados relativos ao esforço e os armazenou em uma planilha do Excel (ver Tabela 6.1). Todos os membros de cada grupo coletavam dados separadamente à medida que realizavam suas atividades como codificar, testar, projetar, etc. No entanto, existia apenas uma planilha oficial por grupo, de responsabilidade do rastreador, que armazenava as informações coletadas por todos os membros.

Tabela 6.1 – Planilha de dados de esforço

Data	Começo	Fim	Pausa	Delta	Iter.	Discipl.	Atividade	Pessoas (login)	Artefatos	Obs.
01/12/2003	13:20	14:50	00:10	01:20	1	XPDRR	Definir req. iniciais	atfs,egs,pst	Story Cards (30 Min)	-

A Tabela 6.1 mostra um exemplo de como os dados foram preenchidos no experimento. Ela mostra que três pessoas (atfs, egs, pst) participaram da atividade de definição inicial dos requisitos dentro da disciplina de código XPDRR. Este código foi definido com base nos nomes das disciplinas de ambos os processos (mostrados na modelagem na Figura 4.4 e na Figura 5.2). A atividade criou os cartões de estória em trinta minutos e o tempo total gasto na atividade (Delta) foi de uma hora e vinte minutos.

Todos os grupos envolvidos no experimento coletaram os dados usando este mesmo formato de planilha e no final do projeto as oito planilhas foram entregues para servir de base para a análise quantitativa. Com base nos dados coletados nas planilhas, pode-se ter noção de questões como:

- Quanto tempo é gasto em cada disciplina de XP e XWebProcess;
- Quanto de esforço adicional as adaptações de XWebProcess representam;

- Quais as vantagens e desvantagens de usar XP ou XWebProcess para o desenvolvimento Web.

Além de coletar os dados com a planilha, um questionário foi elaborado e entregue a todos os alunos, após o término do experimento, com o objetivo de coletar informações para uma análise qualitativa de cada processo. É importante citar que os grupos que usaram XP tiveram treinamento em XWebProcess após o término do experimento e antes de responder ao questionário.

O questionário, mostrado no Apêndice C, procurou obter respostas a respeito da qualidade e da visão das pessoas sobre cada processo utilizado. Isso possibilitou obter feedback sobre questões como:

- A facilidade de entendimento da definição de cada processo;
- Se a definição dos processos através da modelagem está bem estruturada;
- Como esta modelagem facilita no entendimento da dinâmica e da estrutura de cada processo;
- Qual dos processos é mais adequado para o desenvolvimento Web e porquê.

6.3 Análise e interpretação dos resultados

Esta seção mostra a análise quantitativa (Seção 6.3.1) feita com base nos dados coletados nas planilhas e também apresenta a análise qualitativa (Seção 6.3.2) com base nos questionários respondidos por todos os alunos envolvidos no experimento.

6.3.1 Análise quantitativa

A Tabela 6.2 mostra os resultados (esforço em pessoa-hora) obtidos pelos oito grupos envolvidos no experimento. Estes dados foram coletados durante o experimento, que teve a duração de vinte dias, e representam o esforço gasto nas atividades realizadas. O total do esforço foi calculado para cada grupo e é mostrado na Tabela 6.2.

Tabela 6.2 - Esforço gasto no experimento

Esforço gasto (pessoa-hora)							
XP				XWebProcess			
G1	G2	G3	G4	G5	G6	G7	G8
121,03	106,83	98,17	90,67	129,25	125,05	110,67	90,68

Uma forma eficiente de analisar os dados acima, sugerida em [80], consiste em usar o teste t para comparar os dois processos em questão. Tal teste visa comparar se o esforço gasto para desenvolver o sistema utilizando ambas as técnicas é o mesmo e seu objetivo é comprovar a hipótese nula (HN) mostrada abaixo.

Hipótese Nula (HN): O esforço gasto para desenvolver o sistema usando XWebProcess é o mesmo que usando XP, i.e. as médias esperadas são iguais ($\mu_x = \mu_y$).

Hipótese Alternativa (HA): O esforço gasto para desenvolver o sistema usando XWebProcess e XP são diferentes ($\mu_x \neq \mu_y$).

O objetivo do experimento então é verificar se a hipótese nula é verdadeira. Os valores em x representam o esforço gasto pelos grupos de XP (G1, G2, G3 e G4) e os valores em y representam o esforço gasto pelos grupos de XWebProcess (G5, G6, G7 e G8). As fórmulas apresentadas em [80] para realizar o teste t são apresentadas abaixo.

HN: $\mu_x = \mu_y$, i.e. as médias esperadas são iguais.

HA: $\mu_x \neq \mu_y$: Rejeite HN, Se $|t| > t_{\alpha/2,f}$ onde $f = n + m - 2$ é o grau de liberdade. n é o número de observações de x e m de y. α é o nível de significância e $t_{\alpha/2,f}$ é um valor que pode ser encontrado em tabelas estatísticas presentes em livros como em [80].

$$t = \frac{\bar{x} - \bar{y}}{Sp\sqrt{\frac{1}{n} + \frac{1}{m}}}, \text{ onde } Sp = \sqrt{\frac{(n-1)S_x^2 + (m-1)S_y^2}{n+m-2}} \text{ e, } S_x^2 \text{ e } S_y^2 \text{ são as variâncias.}$$

A análise acima foi feita usando uma das ferramentas de análise de dados do Excel para o teste t (duas amostras presumindo variâncias equivalentes). Os resultados da Tabela 6.3 mostram que t = -0,8954 e $t_{\alpha/2,f} = 2,447$ para $\alpha = 5\%$ e f = 6. Portanto, a

hipótese nula (HN) é verdadeira, pois $|t| < t_{\alpha/2,f}$, mostrando que o esforço gasto nos dois processos foi equivalente.

É importante observar que em valores absolutos as equipes de XWebProcess tiveram um esforço maior que as equipes de XP, o que pode ser comprovado pelas médias mostradas na Tabela 6.3 (113,9 em XWebProcess contra 104,1 em XP). Isso pode ser justificado pela existência de um maior número de atividades e artefatos em XWebProcess o que acaba demandando mais esforço. No entanto, a análise estatística provou que este esforço a mais não é significativo, e pelo teste t, pode-se considerar que os dois processos possuem o mesmo desempenho.

Tabela 6.3 - Análise de dados

RESULTADOS GERAIS		
	Equipes de XP	**Equipes de XWebProcess**
Média	104,175	113,9125
Variância	169,8617	303,1819
Número de observações	4	4
RESULTADO DO TESTE T		
Estatística t	-089542	
t crítico bi-caudal ($t_{\alpha/2,f}$)	2,446914	
Conclusão	HN é verdadeira, pois $\|t\| < t_{\alpha/2,f}$	

6.3.2 Análise qualitativa

A análise qualitativa se baseou no questionário apresentado no Apêndice C. Os alunos envolvidos no experimento responderam o questionário após o término do projeto e aqueles que não haviam tido treinamento em XWebProcess (equipes de XP) obtiveram tal treinamento antes de responder ao questionário.

A principal intenção dos questionários foi obter um “feedback” sobre o uso de ambos os processos. Os questionários focaram em questões como: facilidade de entendimento do processo, benefícios, desvantagens e adequação ao desenvolvimento Web. Além disso, foi pedido a cada participante que fizesse uma análise comparativa dos processos. Alguns resultados importantes são mostrados abaixo.

- Todos os alunos se mostraram satisfeitos com o uso do processo (XP e XWebProcess) e demonstraram interesse em usá-los novamente;
- As principais vantagens citadas de se trabalhar com ambos os processos foram: um menor grau de formalismo e menos rigor na documentação em favor de maior comunicação e interação entre a equipe. Isso foi apontado como um fator determinante na velocidade de trabalho da equipe e na redução de dúvidas e ambigüidades. Além disso, a programação em par auxilia no entendimento sobre o sistema e os testes automatizados ajudam a evitar a introdução de erros garantindo a boa qualidade do código;
- 100% das pessoas mencionaram que XWebProcess é mais adequado ao desenvolvimento Web que XP ao responder a questão 5. Elas afirmaram que as disciplinas inseridas e adaptadas contêm elementos (atividades, papéis, artefatos) de extrema importância para o desenvolvimento Web. Algumas atividades como: projetar interface e navegação Web foram consideradas de suma importância para a criação de um sistema Web bem organizado, fácil de navegar e com boa aparência. O uso de prototipação em XWebProcess foi apontado como benéfico para o levantamento dos requisitos e para estabelecer um bom relacionamento com o cliente. A disciplina de testes Web também foi considerada essencial para a verificação de requisitos não funcionais como carga, segurança e concorrência, principalmente em sistemas Web de grande porte.
- Todos os alunos que usaram XWebProcess mencionaram que apesar de ter mais elementos (disciplinas, atividades, artefatos) que XP, usar o processo é mais vantajoso, pois o mesmo é mais adequado para a Web e ainda continua ágil e permitindo a construção de aplicações com uma qualidade melhor;
- 95% dos alunos que usaram XP concordaram que o processo realmente precisa ser adaptado para o desenvolvimento Web conforme feito em XWebProcess;

- Todos os entrevistados mencionaram que a modelagem dos processos usando SPEM ajuda a melhorar o entendimento da sua estrutura e dinâmica e também facilita a sua adaptação para as necessidades da organização e dos projetos.

As observações acima são suficientes para apresentar o conteúdo das respostas sobre as questões 1 e 5 do questionário. Abaixo são apresentados mais dados, obtidos sobre as respostas das outras perguntas (2, 3 e 4).

Facilidade de compreensão (questão 2)

A Figura 6.1 mostra os resultados sobre a definição do processo, ou seja, a facilidade que se tem de entender o processo através da leitura da sua modelagem com SPEM.

A maioria (95%) dos entrevistados que usaram XP e XWebProcess afirmou que o processo é bem definido com suas atividades bem delimitadas e explicadas pela modelagem SPEM. Esta modelagem foi apontada como responsável por facilitar o entendimento das responsabilidades de cada papel e da seqüência que as atividades seguem desde o início até o final do projeto.

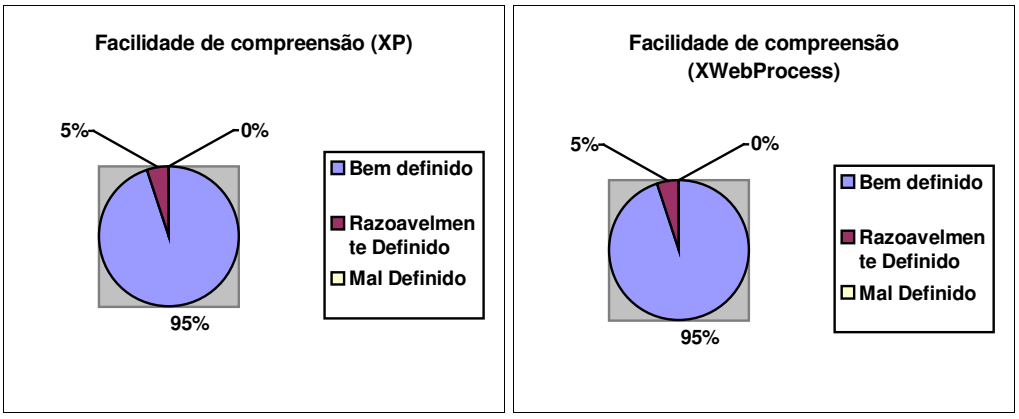


Figura 6.1 - Facilidade de compreensão do processo

Uma minoria (5%) classificou o processo utilizado como razoavelmente definido. Em ambos os casos a justificativa dada foi que não se tem uma idéia muito detalhada da responsabilidade de cada papel e apenas uma idéia razoável da seqüência

das atividades ao longo do projeto. Apesar disso, todos concordaram que a modelagem do processo usando SPEM facilita o seu entendimento.

Visibilidade (questão 3)

A Figura 6.2 mostra os resultados sobre a visibilidade do processo. Esta característica é complementar a anterior e procura analisar se os entrevistados acreditam que o processo apresenta resultados nítidos de modo que o progresso do projeto é visto de forma clara.

Para 90% dos entrevistados de XP e 95% dos entrevistados de XWebProcess o processo é bem visível, pois suas atividades definem de forma clara os artefatos de entrada e saída e os responsáveis por ele e sabe-se quais artefatos são essenciais ou não. Segundo eles, isso é facilitado pela boa descrição do processo usando SPEM, principalmente através da modelagem dinâmica com diagrama de atividades, o que possibilita ter uma visão nítida do início e fim das atividades e de quais artefatos são fornecidos como entrada ou produzidos como saída.

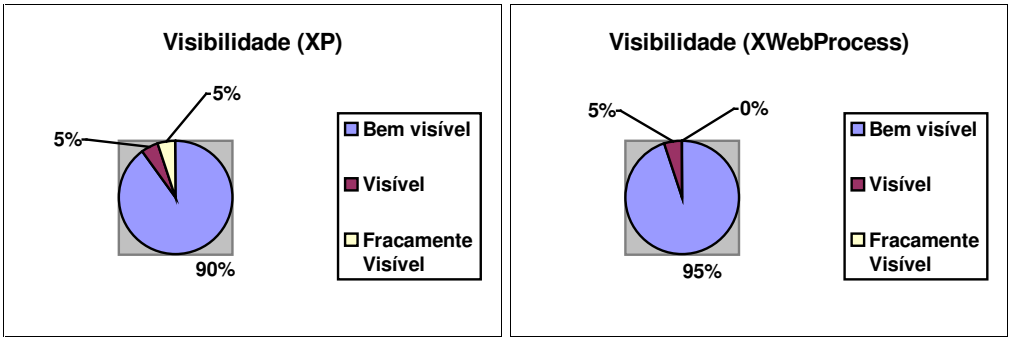


Figura 6.2 – Visibilidade do Processo

Já 5% dos entrevistados de XP e XWebProcess classificaram o processo como visível, pois as atividades do processo definem de forma clara os artefatos de entrada e saída e os responsáveis por ele, porém não se sabe a respeito da sua importância (descartável ou não). Para XP, 5% dos entrevistados classificaram o processo como fracamente visível e apontaram que o processo não delimita de forma nítida o fluxo de suas atividades bem como quais artefatos são essenciais ou não.

Facilidade de manutenção (questão 4)

A Figura 6.3 mostra os resultados sobre a facilidade de manutenção no processo. Este item visa medir se o processo pode evoluir para refletir os requisitos mutáveis da organização ou melhorias de processo identificadas. Ou seja, mede a facilidade que se tem de alterar o processo para adaptá-lo a um requisito novo da organização ou de um domínio em particular.

Para 85% dos entrevistados de XP e 95% dos entrevistados de XWebProcess o processo é fácil de ser alterado, pois devido a sua boa definição fica mais fácil fazer adaptações nele para atender a necessidades específicas de projeto e da organização.

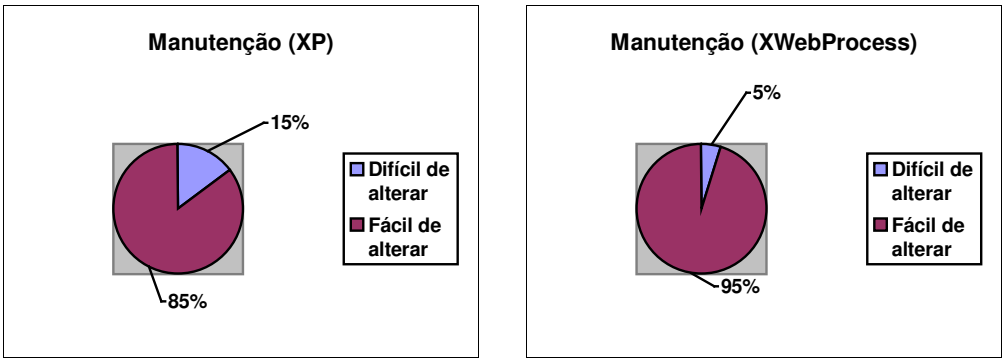


Figura 6.3 – Facilidade de manutenção do processo

Já 15% acreditam que XP é um processo que não é fácil de ser modificado enquanto 5% compartilham da mesma opinião para XWebProcess.

É importante mencionar que os entrevistados que avaliaram as questões anteriores (compreensão, visibilidade e manutenção) fizeram esta avaliação com base na definição de XP e XWebProcess apresentadas nesta dissertação, ou seja, descrita de forma textual e modelada com SPEM.

Todos os entrevistados que usaram ambos os processos apontaram a definição do processo com SPEM como fator responsável pela sua boa qualidade. Os entrevistados de XP apontaram que sem a modelagem feita, ou seja, se baseando apenas na literatura comum de XP, fica difícil compreender a dinâmica do processo e saber como adaptar o processo para uma situação em particular. Os entrevistados de

XWebProcess também compartilharam da opinião de que a modelagem facilita o entendimento do processo e também a sua aplicação na prática. Então, conclui-se que a modelagem feita fez com que os critérios qualitativos anteriores fossem avaliados positivamente, apontando um bom nível de qualidade nos dois processos.

Resumo geral da avaliação qualitativa

As respostas para obtidas para os entrevistados dos dois processos (XP e XWebProcess) são resumidas na Tabela 6.4.

Tabela 6.4 – Análise qualitativa de XP e XWebProcess (questões 2, 3 e 4 do questionário)

	Critério Qualitativo							
	Facilidade de Compreensão			**Visibilidade**			**Manutenção**	
Processo	Bem definido	Razoav. Definido	Mal Definido	Bem visível	Visível	Fracamente Visível	Fácil de alterar	Difícil de alterar
XP	95%	5%	0%	90%	5%	5%	85%	15%
XWeb Process	95%	5%	0%	95%	5%	0%	95%	5%

O objetivo das análises dos critérios qualitativos das questões 2, 3 e 4 do questionário não eram o de comparar XP com XWebProcess, mas sim o de obter informações sobre como a modelagem de cada processo teve impacto nos critérios qualitativos apresentados aqui. A comparação entre os dois processos no questionário foi feita na questão 5 e já foi apresentada anteriormente. Além disso, a comparação quantitativa dos dois processos foi apresentada na Seção 6.3.1.

6.4 Considerações Finais

Apesar de existirem muitos resultados interessantes mostrados na indústria e na academia sobre processos ágeis, poucos investigam o esforço e o tempo gasto sob uma perspectiva experimental. Os resultados mostrados apontam que os projetos foram entregues com sucesso, ou seja, dentro do prazo e orçamento, mas não apontam porque estes processos ágeis são eficientes e qual esforço gasto dentro das suas atividades.

Este capítulo procurou apresentar o experimento conduzido para a validação de XWebProcess. O estudo procurou comparar o esforço gasto para o desenvolvimento de um mesmo sistema Web utilizando dois processos ágeis: XP e XWebProcess.

O experimento indicou que XWebProcess é uma boa alternativa para o desenvolvimento de aplicações Web de forma ágil. Foi provado que o esforço gasto em ambos os processos foi o mesmo segundo o critério do teste t. Isso significa que apesar de XWebProcess ter mais elementos (disciplinas, artefatos, papéis) isso não representa um aumento significativo no esforço gasto.

Além disso, o questionário realizado nos ajudou a obter informações importantes a respeito da qualidade dos dois processos no que diz respeito às suas descrições (modelagem com SPEM) e uso prático no projeto. Com isto pode-se ver que a modelagem feita facilitou o entendimento e uso do processo pelos alunos. Os entrevistados também mencionaram que a modelagem ajuda a entender melhor a estrutura e dinâmica do processo e torna mais fácil fazer futuras adaptações para se adequar ao ambiente da organização e características de projetos.

Um ponto negativo do experimento foi o fato da maioria dos alunos não terem experiência prévia com as tecnologias Web escolhidas: JSP e Servlets. A maioria deles já tinha experiência com outras tecnologias Web, mas achamos que seria melhor fixarmos a tecnologia para não se ter um impacto maior no experimento. Apesar disso, todos foram capazes de aprender e entregaram o sistema no prazo estabelecido. Dois fatores, apontados pelos alunos, que contribuíram para este rápido aprendizado foram a programação em pares e a constante comunicação entre os membros da equipe.

Portanto, o experimento conduzido foi de grande utilidade para a validação do processo XWebProcess. Foi mostrado que este processo, apesar de ser uma adaptação de XP com mais elementos, continua a ser um processo ágil cujo esforço é equivalente a XP e ainda mais adequado sob o ponto de vista Web, possibilitando a construção de aplicações Web de boa qualidade e com agilidade.

7
Conclusões

Este capítulo mostra as conclusões sobre esta dissertação. A Seção 7.1 descreve as principais contribuições deste trabalho. A Seção 7.2 aponta as maiores dificuldades encontradas. Finalmente, a Seção 7.3 aponta as perspectivas de trabalhos futuros.

7.1 Principais contribuições

Neste trabalho foi apresentado um processo ágil para o desenvolvimento de aplicações Web. XWebProcess foi desenvolvido com base em XP e procurou levar em consideração as características especiais de aplicações Web que precisam ser tratadas por um processo de desenvolvimento.

Extreme Programming é a metodologia ágil mais popular da atualidade e suas características de interação constante, pouca documentação, pequenas equipes e projeto simples favorecem a dinâmica inerente ao desenvolvimento para a Web, onde a questão de prazo de entrega é fundamental para a estratégia da empresa.

XWebProcess foi criado procurando respeitar os princípios e práticas de XP para que sua característica ágil fosse mantida. As adaptações realizadas procuraram tratar as principais questões referentes ao desenvolvimento Web, como interfaces gráficas mais complexas, projeto navegacional e testes de requisitos não funcionais. No entanto, teve-se o cuidado para que tais adaptações não "amarrassem" o processo a nenhuma técnica, tecnologia ou método em particular.

A descrição de XWebProcess foi feita utilizando a linguagem SPEM que permitiu representar o processo usando diagramas da UML, tais como diagramas de

atividades e de classes. Isto facilitou não só o entendimento do processo, mas também as próprias adaptações feitas.

O experimento conduzido para a validação de XWebProcess mostrou que as adaptações feitas não representaram um aumento significativo no esforço empregado pelas equipes que a utilizaram. Apesar de conter mais elementos que XP, o questionário aplicado evidenciou que XWebProcess é adequado para o desenvolvimento Web, pois suas adaptações tratam de questões de fundamental importância para este tipo de aplicação.

Portanto, acreditamos que XWebProcess tenha atendido ao seu principal objetivo inicial que era o de prover um meio adequado e eficiente para a construção de aplicações Web de forma rápida e com qualidade.

7.2 Dificuldades encontradas

As principais dificuldades encontradas se deram principalmente devido à existência de pouca pesquisa relacionando os assuntos de engenharia de software para a Web e processos ágeis. Alguns destes trabalhos são citados na próxima seção.

A primeira dificuldade encontrada foi delimitar o que seria efetivamente tratado por XWebProcess e como este processo seria estruturado. Optamos por nos concentrar nas disciplinas relativas ao desenvolvimento de software como: requisitos, análise e projeto, implementação e testes. A nossa escolha foi dar uma visão geral destas disciplinas sem detalhá-las em demasia, o que poderia tornar o processo pesado.

Procuramos nos basear em XP devido ao sucesso que este processo vem alcançando ultimamente. Outro desafio encontrado foi a escolha de SPEM como mecanismo de abstração para auxiliar na definição e estruturação de XWebProcess. Existe ainda pouca literatura existente sobre SPEM e tivemos que nos basear basicamente na sua definição formal apresentada em [48].

A última dificuldade encontrada foi o planejamento, execução e posterior análise do experimento de validação. Foi bastante complicado conseguir gerenciar as oito equipes, que totalizavam quarenta pessoas, e verificar se os processos estavam sendo seguidos de forma correta e se os dados sobre o experimento estavam condizentes com a realidade obtida na prática. Foram necessárias algumas reuniões com os alunos fora do horário de aula, para verificar o andamento do projeto e isto acabou exigindo um pouco mais de esforço tanto da nossa parte quanto da parte dos alunos.

7.3 Trabalhos futuros

As disciplinas de XWebProcess correspondem a uma extensão das disciplinas de XP onde algumas foram modificadas e outras inseridas. Estas disciplinas foram descritas em um alto nível de abstração sem se preocupar com muitos detalhes. Um dos principais objetivos era que o processo não ficasse muito dependente de uma tecnologia, método ou ferramenta em particular e nem pesado demais.

Um ponto de vista interessante que pode ser tratado em futuros trabalhos seria um detalhamento maior das disciplinas existentes em XWebProcess e a inserção de disciplinas para tratar de questões que não foram contempladas como gerência de configuração e gerência de projetos.

Tais extensões podem se beneficiar da modelagem feita com SPEM e adicionar outros elementos, semelhante ao que foi feito nesta dissertação com a modelagem de XP. É importante também tentar preservar a agilidade do processo e sua independência em relação a tecnologias específicas.

No que diz respeito à validação do processo pode ser interessante conduzir experimentos mais longos para comparar XWebProcess com outros processos ágeis e até com processos mais pesados como o RUP.

Referências Bibliográficas

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

[2] Simon, E., ***Distributed Information Systems***, McGraw-Hill, 1996.

[3] Berners-Lee, T., ***WWW: Past, Present and Future***, IEEE Computer, October, 1996.

[4] McDonald, A. and Welland, R., ***Agile Web Engineering (AWE) Process***, Department of Computing Science Technical Report TR-2001-98, University of Glasgow, 2001.

[5] S. Hansen, Y. Deshpande, S. Murugusan and A. Ginige, ***Web Engineering: A New Discipline for Development of Web-based Systems,*** First Workshop on Web Engineering in International Conference on Software Engineering, Los Angeles, USA, 16-22 May 1999.

[6] ***Web Engineering Home Page***, http://www.aeims.uws.edu.au/WebEhome/, último acesso em 01/01/2004.

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

[8] Pressman, R.S., ***Can Internet-Based Applications be Engineered?***, IEEE Software, September – October 1998, pp. 104-110.

[9] Pressman, R.S., ***What a Tangled Web We Have***, IEEE Software, January – February 2000, pp. 18-21.

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

[11] Souza, R.A.C, ***Uma Extensão do Fluxo de Análise e Projeto do RUP para o Desenvolvimento de Aplicações Web***, Dissertação de Mestrado, Centro de Informática - UFPE, 2002.

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

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

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

[15] Rossi, G., ***The Object Oriented HiperMedia Design Method (OOHDM)***, tese de Doutorado, Departamento de informática, PUC-Rio, 1996.

[16] Chen, J., and Chang, S., ***An object-oriented method for software mantainance***, JOOP, January 1994, Vol6 N. 8, pp 46-51.

[17] Izakowitz, T., Stohr, E., and Balasubramaniam, P., ***RMM: A methodology for structured hypermedia design***, Communications of the ACM, October 1995, pp 34-44.

[18] Schwabe, D. and Rossi, G., ***The Object-Oriented Hypermedia Design Model,*** In Communications of the ACM, volume 38(8), pages 45-46, 1995.

[19] Fowler, M., ***The New Methodology***, MartinFowler.com, May 2002, http://www.martinfowler.com/articles/newMethodology.html, último acesso em 01/03/2004.

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

[21] ***Object oriented Process, Environment and Notation (OPEN) Web site***, http://www.open.org.au/, último acesso em 01/03/2004.

[22] Jeffries, R.E., ***XProgramming.com: an Extreme Programming Resource***, http://www.xprogramming.com/, último acesso em 01/03/2004.

[23] ***Dynamic System Development Method (DSDM) Web Site***, http://www.dsdm.org, último acesso em 01/03/2004.

[24] ***Crystal Web Site***, http://www.crystalmethodologies.org/, último acesso em 01/03/2004.

[25] ***Java 2 Enterprise Edition (J2EE) Web Site***, http://java.sun.com/j2ee, último acesso em 01/03/2004.

[26] ***Microsoft.NET Web Site***, http://www.microsoft.com/net/, último acesso em 01/03/2004.

[27] Vawter, C., and Roman, E., ***J2EE vs. Microsoft.NET A comparison of building XML-based web services***, June 2001, http://www.middleware-company.com/

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

[29] Jeffries, R., Anderson, A., and Hendrickson, C., ***Extreme Programming Installed***, Addison-Wesley, 2001.

[30] Highsmith, J., ***Extreme Programming***, February 2000, http://www.cutter.com, último acesso em 01/03/2004.

[31] Fowler, M., ***Refactoring: Improving the Design of Existing Code***, Addison-Wesley, 1999.

[32] ***The American Heritage Dictionary***, Second College Edition, 1991.

[33] ***Extreme Programming Website***, http://www.extremeprogramming.org/, último acesso em 01/03/2004.

[34] ***The Oficial Agile Modelling Web Site***, http://www.extreme-modeling.com/, último acesso em 01/03/2004.

[35] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., **Design Patterns Elements of Reusable Object-Oriented Software**, Addison-Wesley Professional Computing Series, 1995.

[36] ***WebSite da UML da OMG***, www.omg.org/uml/, último acesso em 01/03/2004.

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

[38] Pastor, O., Fons, J., Abrahão, S., and Rámon, S., ***Object Oriented Conceptual Models for WEB Applications,*** 4th Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS´2001), San Jose, Costa Rica, 2001.

[39] Henderson-Sellers, B., Lowe, D., and Haire, B., ***OPEN Process Support for Web Development,*** Annals of software Engineering, vol. 13, pp 163-202, 2002.

[40] Soares, S., Laureano, E. and Borba, P., **Implementing distribution and persistence aspects with AspectJ**, In 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages and Applications, OOPSLA'2002, pages 174-190, Seattle, USA, 4th-8th November 2002.

[41] Soares, S. and Borba, P., **Controle de concorrência com Java e bancos de dados relacionais**, In *V Brazilian Symposium on Programming Languages*, pages 252-267, Curitiba, Brazil, 23rd-25th May 2001.

[42] Coulouris, G., Dollimore, J., and Kindberg, T., ***Distributed Systems - Concepts and Design,*** Addison-Wesley, 2001.

[43] Massoni,T., Alves, V., Soares, S., and Borba, P., ***PDC: The persistent data collections patter,*** *In* First Latin American Conference on Pattern Languages of Programming, Rio de Janeiro, Brazil, 3th-5th October 2001.

[44] Pressman, R., ***Software Engineering: A practioner´s approach***, Third edition, McGraw-Hill, 1992.

[45] JavaWorld Magazine, ***Rumble in the jungle: J2EE versus .Net -How do J2EE and Microsoft's .Net compare in enterprise environments***, Junho de 2002.

[46] A Rational Software and Context Integration white paper, ***Building Web Solutions with the Rational Unified Process: Unifying the Creative Design Process and the Software Engineering Process***, 1999.

[47] Sommerville, I., ***Engenharia de Software,*** 6ªEdição, Addison Wesley, 2003.

[48] Object Management Group, ***Software Process Engineering Metamodel Specification,*** Versão 1.0, Especificação adotada como padrão da OMG em 14 de Novembro de 2002.

[49] McDonald, A., and Welland, R., ***Web Engineering in Practice,*** Proceedings of the Fourth WWW10 Workshop on Web Engineering, Page(s): 21-30, May 2001.

[50] Lowe, D., and Eklund, J., ***Client Needs and the Design Process in Web Projects***, Journal of Web Engineering, vol. 1, p. 23-36, 2002, Rinton Press.

[51] Acuña, S., and Ferré, X., ***Software Process Modelling***, Proceedings of The 5th World Multiconference on Systemics, Cibernetics and Informatics (SCI 2001). Orlando, Florida,USA. 1-6, July 2001.

[52] Conradi, R., Fernström, C., and Fuggetta, A., ***Concepts for Evolving Software Process***, in Software Process Modelling and Technology chapter 2, Research Studies Press, 1994.

[53] Humphrey, W., ***A Discipline for Software Engineering***, Addison Wesley, 1995.

[54] Cockburn, A., ***Agile Software Development,*** The Agile Software Development Series, Addison Wesley, 2001.

[55] Ambler, S., ***Different Projects Require Different Strategies,*** http://www.agiledata.org, publicado em Dezembro de 2002.

[56] Lindval M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L., and Zelkowitz, M., ***Empirical Findings in Agile Methods,*** Lecture Notes in computer Science, Volume 2418, pp 0197-0207, 20 de Setembro de 2002.

[57] ***Agile Alliance Website***, http://www.agilealliance.org, último acesso em 01/03/2004.

[58] Rumpe, B., and Schröder, A., ***Quantitative Survey on Extreme Programming Projects***, in Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, XP2002, May 26-30, Alghero, Italy, pg. 95-100, 2002.

[59] ***Extreme Programming and Web Development Process*** – Applying (some of) the Values, Principles and Practices of Extreme Programming to the Overall Web Development Process, disponível em http://www.evolt.org/article/ Extreme_Programming_ and_Web_Development_Process/20/16220/.

[60] Wallace, D., Raggett, I., and Aufgang, J., ***Extreme Programming for Web Projects (XP Series)***, Addison-Wesley, 2002.

[61] ***JUnit Web site***, http://*www.junit.org*, último acesso em 01/03/2004.

[62] ***JFunc Web site***, http://jfunc.sourceforge.net, último acesso em 01/03/2004.

[63] **JWebUnit website**, http://jwebunit.sourceforge.net, último acesso em 01/03/2004.

[64] Gelowitz, C., et al, ***Real-Time Extreme Programming***, in Fourth International Conference on Extreme Programming and Flexible Processes in Software Engineering, XP2003, May 25-29, Genova, Italy, 2003.

[65] Heinecke, H., Noack, C., and Schweizer, D., ***Constructing Agile Software Processes,*** In Proceedings of Extreme Programming Conference (XP2002), Alghero, Sardinia, Italy 2002.

[66] Curtis, B., Kellner, M., Over, J., ***Process Modeling,*** Communications of The ACM 35, 9 (Setembro de 1992) 75-90.

[67] Heimann, P., Joeris, G., Krapp, C., and Westfechtel, B., ***DYNAMITE: Dynamic Task Nets for Software Process Management,*** Proceedings of ICSE ´18, páginas 331-341, Berlin, Março de 1996.

[68] Jaccheri, M., Picco, G., and Lago, P., ***Eliciting software process models with the E3 language,*** em ACM Transactions on Software Engineering and Methodology (TOSEM) Outubro de 1998 Volume 7 Issue 4

[69] ***The Object Management Group Web Site*** – http://www.omg.org, último acesso em 01/03/2004

[70] ***IBM Web Design Guidelines***, http://www.ibm.com/ibm/easy/eou_ext.nsf/Publish/ 572PV, último acesso em 01/03/2004.

[71] Dolog, P., and Bielikova, M., ***Hypermedia modelling using UML,*** In Proc. of ISM´2002 – Information Systems Modeling, Roznov pod Radhostem, Czech Republic, April 2002.

[72] Ceri, S., Fraternali, P., and Bongio, A., ***Web Modeling Language (WebML): a modeling language for designing web sites,*** Computer Networks and ISDN Systems, 33(1-6):137-157, June 2000.

[73] Grazotto, F., and Paolini, P., ***HDM - a model-based approach to hypertext application design,*** ACM Transactions on information systems, Vol 11, No. 1, pp 1-26, Jan. 1993.

[74] T. Isakowitz, E. Stohr, and P. Balasubramanian, ***RMM: A Methodology for Structuring Hypermedia Design***, Communications of the ACM, Vol. 38, No. 8, pp. 34-44, August 1995.

[75] Pfleeger, S. L., ***Design and Analysis in Software Engineering – Part1: The Language of Case Studies and Formal Experiments***, Software Engineering Notes, vol 19, no 1, October 1994, pages 16-20.

[76] Pfleeger, S. L., ***Experimental Design and Analysis in Software Engineering – Part2: How to Set Up an Experiment***, Software Engineering Notes, vol 20, no 1, January 1995, pages 22-26.

[77] Pfleeger, S. L., ***Experimental Design and Analysis in Software Engineering – Part3: Types of Experimental Design***, Software Engineering Notes, vol 20, no 2, April 1995, pages 14-16.

[78] Pfleeger, S. L., ***Experimental Design and Analysis in Software Engineering – Part4: Choosing an Experimental Design***, Software Engineering Notes, vol 20, no 3, July 1995, pages 13-15.

[79] Pfleeger, S. L., ***Experimental Design and Analysis in Software Engineering – Part5: Analyzing the Data,*** Software Engineering Notes, vol 20, no 5, December 1995, pages 14-17.

[80] Wohlin, C., et al., ***Experimentation in Software Engineering: An Introduction***, Kluwer Academic Puslishers, 2000.

[81] Basili, V., et al., ***Experimentation in Software Engineering,*** IEEE Transactions on Software Engineering, Vol. SE-12, No. 7, July 1986.

[82] Seaman, C., ***Qualitative Methods in Empirical Studies of Software Engineering,*** IEEE Transactions on Software Engineering, Vol. 25, No. 4, July/August 1999.

[83] McDonald A. and Welland R., ***Agile Web Engineering (AWE) Process: Multidisciplinary Stakeholders and Team Communication,*** J. M. Cueva Lovelle et al. (Eds.): Third International Conference on Web Engineering, ICWE 2003, LNCS 2722, Page(s): 515-518, July 2003.

[84] Sampaio, A., ***XWebProcess Website,*** www.cin.ufpe.br/~atfs/xwebprocess

Apêndice A – O meta-modelo SPEM

O Software Process Engineering Metamodel (SPEM) [48] surgiu como resposta a uma solicitação (Request for Proposal) da OMG por um meta-modelo para processos de software. Esta solicitação despertou o interesse de grandes empresas e pesquisadores da área de desenvolvimento de software que contribuíram para a criação da primeira versão de SPEM, adotada como padrão formal da OMG em Novembro de 2002. Algumas empresas e pesquisadores que contribuíram para a criação da versão 1.0 de SPEM foram:

- **Empresas:** Adaptive Ltd., Alcatel, Computer Associates, France Telecom, Fujitsu/DMR, IBM, Q-Labs, Rational Software, Siemens, SOFTEAM, Toshiba, Unisys e Valtech;
- **Pesquisadores:** Brian Henderson-Sellers, Philippe Kruchten, Craig Larman e outros.

SPEM é um meta-modelo que pode ser usado para descrever um processo concreto de desenvolvimento de software ou uma família de processos relacionados. SPEM adota uma abordagem orientada a objetos para modelar processos e usa a UML como notação. A Figura A.1 descreve a arquitetura de quatro níveis de modelagem definida pela OMG e respeitada por SPEM.

O processo real de produção, conforme instanciado para um projeto específico, está no nível M0. No nível M1 encontra-se a definição do processo que foi instanciado em M0 como, por exemplo, RUP, OPEN ou XP. O foco de SPEM está no meta-modelo que se encontra no nível M2 e serve como um template para o nível M1. A especificação de SPEM está estruturada como um perfil UML (UML profile) e provê um meta-modelo completo baseado no MOF que se encontra no nível M3. Essa

abordagem facilita a utilização e construção de ferramentas para UML e ferramentas baseadas em MOF.

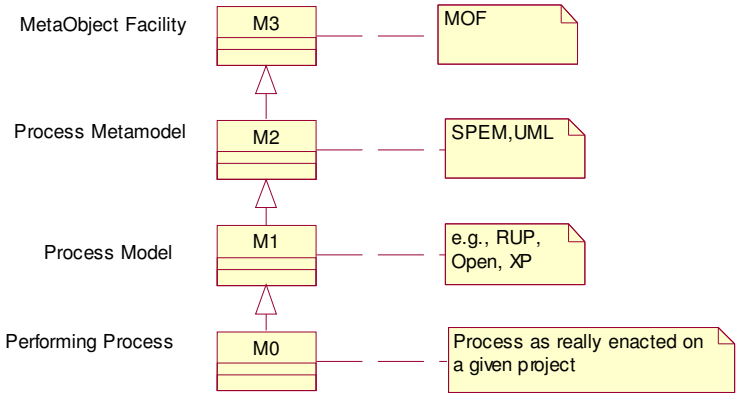


Figura A.1 – Níveis de modelagem definidos em [48]

A elaboração de SPEM teve, desde o início, o objetivo de abranger uma vasta gama de processos de desenvolvimento de software já existentes ao invés de excluí-los, devido ao excesso de elementos e restrições que poderiam existir na sua definição. SPEM é definido como um meta-modelo, e como um perfil UML, o que permite que seus usuários (modeladores de processo) usem a UML e os estereótipos definidos em SPEM como notação.

SPEM é construído pela extensão de um subconjunto do meta-modelo da UML 1.4. Esse subconjunto da UML é chamado de "SPEM_Foundation". Além disso, o modelo SPEM possui o pacote "SPEM Extensions" que adiciona as construções e semânticas requeridas para a engenharia de processos de software e que depende de elementos de "SPEM_Foundation". A Figura A.2 mostra o relacionamento entre esses pacotes.

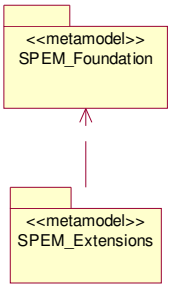


Figura A.2 – Pacotes de SPEM [48]

O pacote SPEM_Foundation é subdivido em seis pacotes:

- **SPEM_Foundation::Data_Types –** Este pacote é um subconjunto do pacote Data_Types da UML 1.4 e contém as definições dos seguintes tipos de dados: Integer, UnlimitedInteger, String, AggregationKind, Boolean, Boolean, ParameterDirectionKind, PseudoStateKind, Name, Multiplicity e MultiplicityRange;
- **SPEM_Foundation::Core –** Este pacote é estruturado de forma similar ao pacote Core da UML 1.4 e contém os elementos de modelagem base do meta-modelo como, por exemplo, Relationship, AssociationEnd, Parameter, Constraint, etc;
- **SPEM_Foundation::Actions –** Este pacote é um subconjunto do pacote Common_Behaviour da UML 1.4 e define os elementos Action, CallAction e Operation;
- **SPEM_Foundation::State_Machines –** Este pacote é um subconjunto do pacote State_Machines da UML 1.4 e define elementos que representam o conceito de máquina de estado como Transition, State, Action, etc;
- **SPEM_Foundation::Activity_Graphs –** Este pacote é um subconjunto do pacote Activity_Graphs da UML 1.4 e define elementos de modelagem de diagrama de atividades como State, Classifier, etc;
- **SPEM_Foundation::Model_Management** – Este pacote é um subconjunto do pacote Model_Management da UML 1.4 e mostra que em SPEM todos os elementos têm visibilidade pública e que os elementos importados para os pacotes não podem ser renomeados.

As definições são importantes para compreender o que o pacote SPEM_Foundations possui. Os diagramas contendo todos os elementos de cada um dos seis pacotes acima podem ser encontrados na especificação de SPEM e são importantes principalmente para os interessados em construir ferramentas que dêem suporte a

SPEM. Para o objetivo deste trabalho, que é apenas o de usar SPEM para produzir modelos, a descrição detalhada de tais pacotes é desnecessária.

O pacote SPEM_Extensions visto na Figura A.2 adiciona as construções e semânticas para a engenharia de processos de software. A Figura A.3 mostra a estrutura interna deste pacote em termos de seus subpacotes e mostra as dependências entre estes e os pacotes de SPEM_Foundations. Os cinco subpacotes de SPEM_Extensions são: BasicElements, Dependencies, ProcessStructure, ProcessComponents e ProcessLifecycle. Estes pacotes são detalhados a seguir.

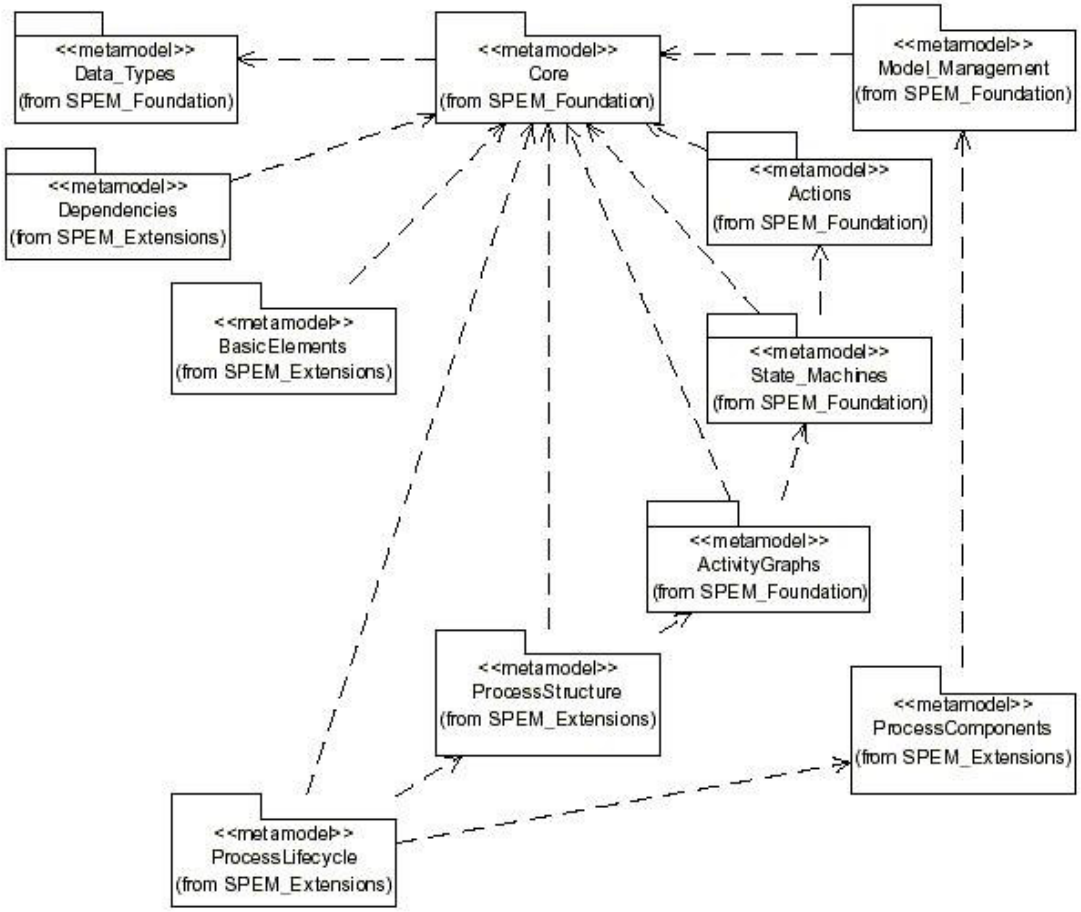


Figura A.3 – Estrutura de Pacotes de SPEM [48]

Pacote BasicElements

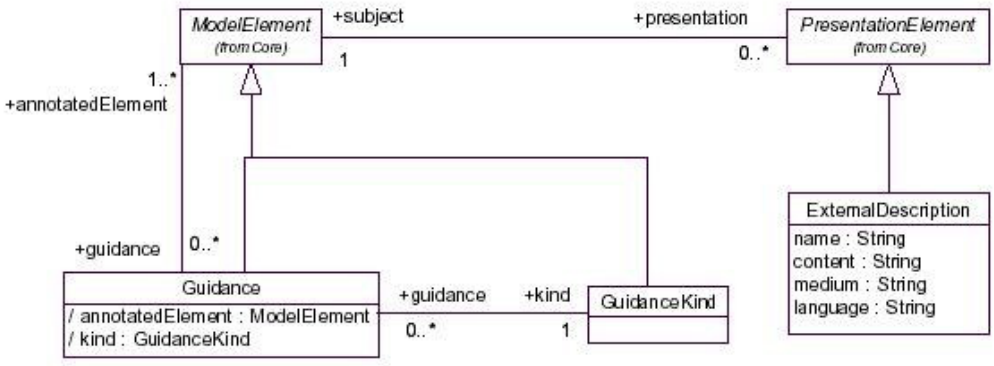


Figura A.4 – Pacote Basic Elements [48]

- **ExternalDescription** contém uma descrição adequada do ModelElement para o leitor da descrição do processo. O atributo content, por exemplo, contém uma descrição em linguagem natural do ModelElement.
- Elementos **Guidance** podem ser associados com ModelElements para prover algum tipo de informação mais detalhada. Exemplos de Guidance podem ser: Guidelines, Techniques, Metrics, Tool Mentors, Checlists, Templates, etc.
- **GuidanceKind** é um elemento de SPEM que procura definir várias categorias de Guidance como uma forma de padronização, visto que processos distintos podem chamar o mesmo conceito por nomes distintos como técnica (technique) ou método (method).

Pacote Dependencies

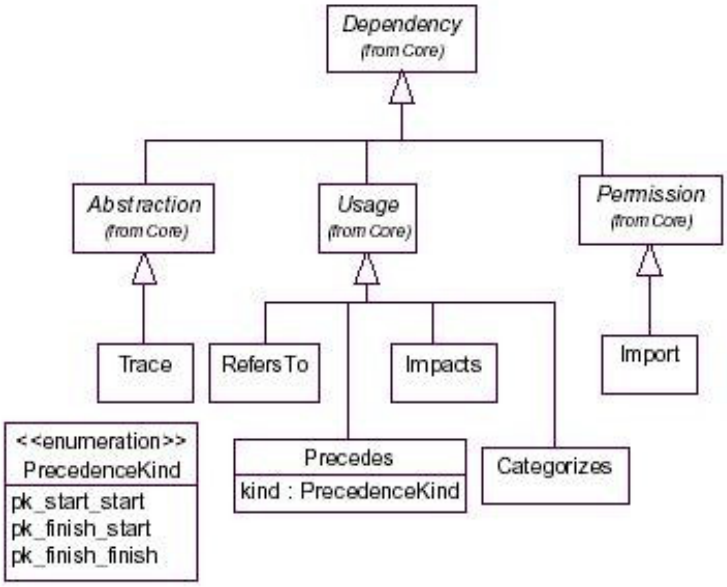


Figura A.5 – Pacote Dependencies [48]

- **Categorizes** é quando um pacote depende de um elemento individual de outro pacote podendo associar tal elemento a múltiplas categorias.
- **Impacts** representa que um WorkProduct quando modificado pode impactar em outro WorkProduct.
- **Import** denota que o conteúdo do pacote alvo é adicionado ao namespace do pacote fonte. É semelhante ao Import da UML - só difere porque em SPEM todos os elementos têm visibilidade pública.
- **Precedes** atua em relação a atividades (Activity ou WorkDefinition) para representar seqüência.
- **RefersTo** atua em relação a elementos do processo para garantir que eles estejam incluídos no mesmo ProcessComponent.
- **Trace** atua em relação a Workdefinitions e serve para rastrear requisitos e mudanças entre os modelos.

Pacote Process Structure

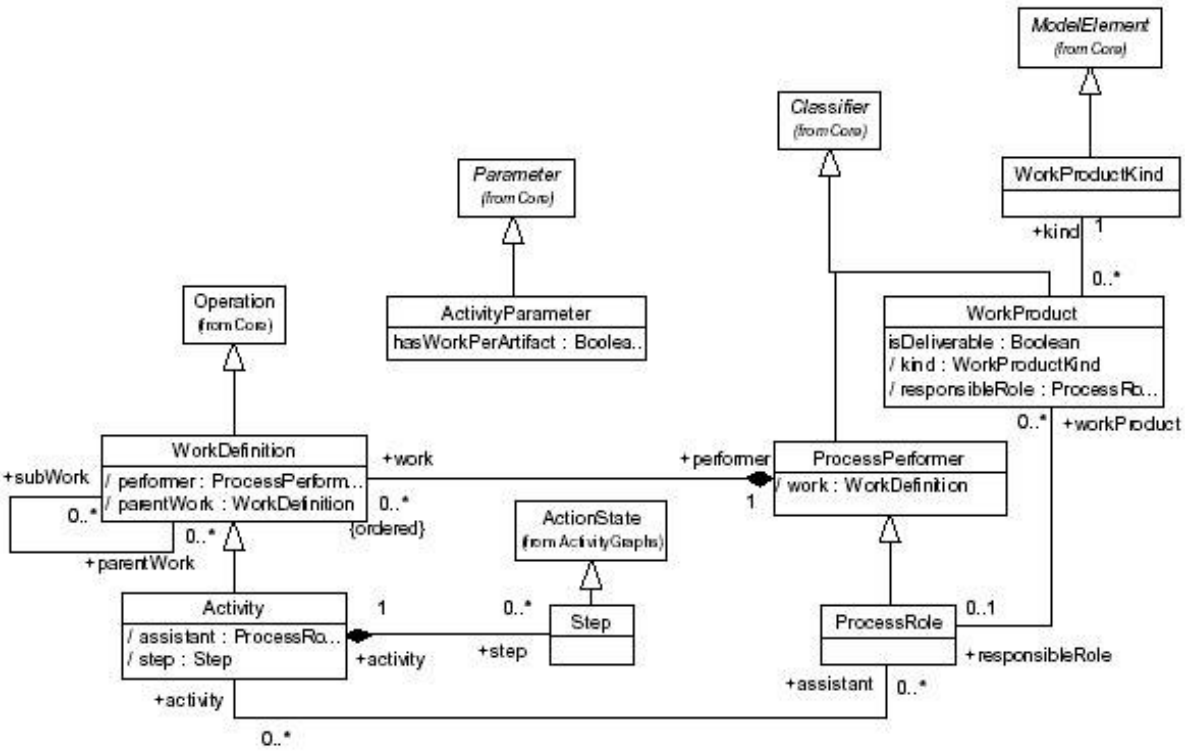


Figura A.6 – Pacote Process Structure [48]

- Um **WorkProduct** (artefato) é algo produzido, consumido ou modificado por um processo. Pode ser um documento, modelo, código fonte, etc.
- Um **WorkProductKind** descreve uma categoria de WorkProduct como um documento texto, modelo UML, biblioteca de código, etc.
- Um **WorkDefinition** é um tipo de operação que descreve o trabalho realizado em um processo e sua principal subclasse é Activity. Seus inputs e outputs são referidos via AcitivityParameter.
- **Activity** é a principal subclasse de WorkDefinition e descreve um subconjunto de trabalho realizado por um ProcessRole, ou seja, as tarefas (tasks), operações e ações que são realizadas ou assistidas por um

ProcesRole. Uma atividade pode consistir de elementos atômicos chamados **Steps**.

- **ProcessPerformer** define o realizador de um conjunto de WorkDefinitions em um processo. ProcessPerformer possui uma subclasse **ProcessRole** que define as responsabilidades sobre WorkProducts específicos informando se o papel é assistente ou responsável.

Pacote Process Components

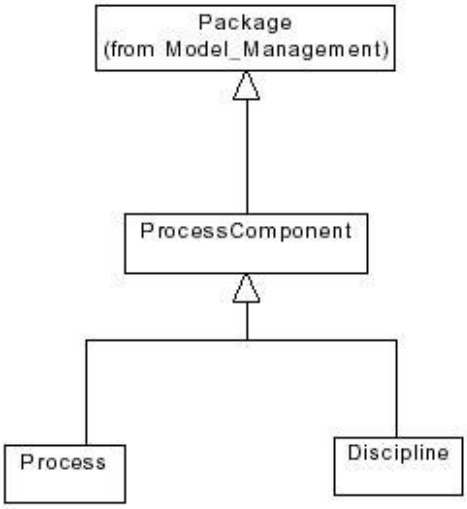


Figura A.7 – Pacote Process Components [48]

- Assim como em UML, um **Package** é um container que pode possuir ou importar elementos de definição do processo.
- Um **ProcessComponent** é um subconjunto do processo que é internamente consistente e pode ser reusado com outros componentes para montar um processo completo.
- Um **Process** é um ProcessComponent que sozinho representa um processo completo, ou seja, não precisa ser composto com outros como o ProcessComponent.

- **Discipline** é um ProcessComponent que representa um conjunto de WorkDefinitions, Roles e WorkProducts sob um ponto de vista comum como análise e projeto, teste, implementação, etc.

Pacote Process Lifecycle

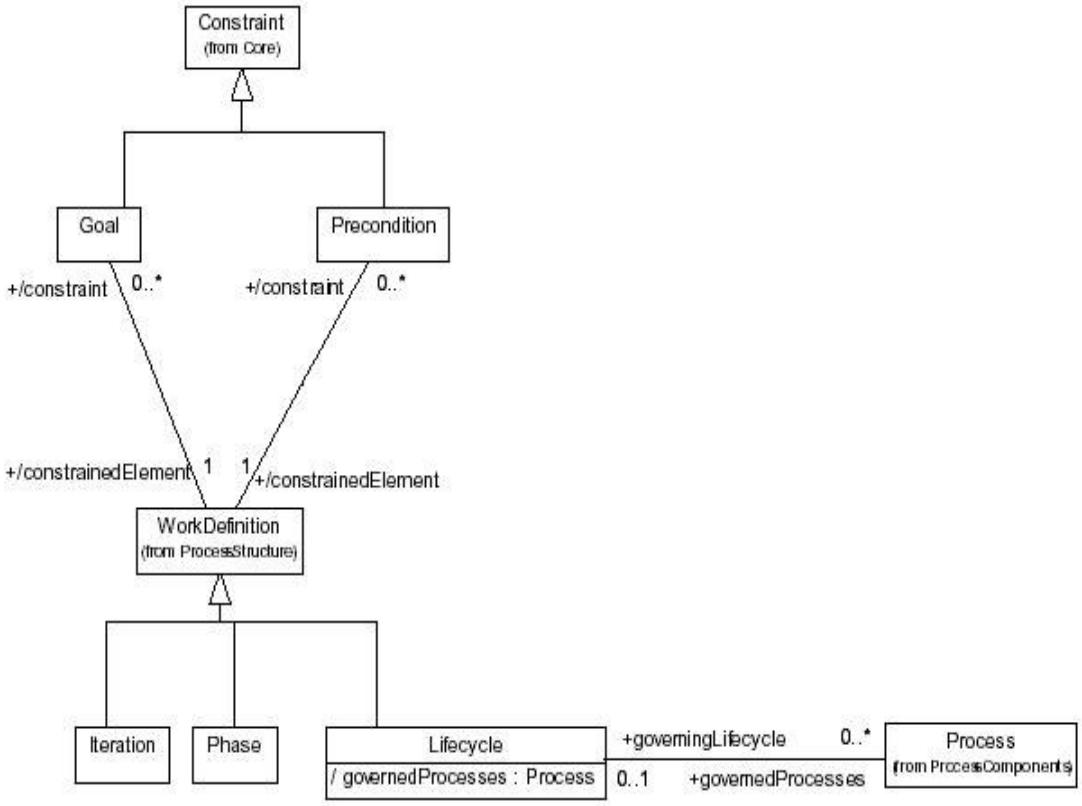


Figura A.8 – Pacote Process Lifecycle [48]

- **Phase** é uma especialização de WorkDefinition cuja precondição (Precondition) define o critério de entrada e seu objetivo (Goal) define o critério de saída.
- **Lifecycle** representa uma seqüência de fases (Phases) que procuram atingir um objetivo específico. Representa o comportamento de um processo a ser instanciado em um projeto específico.
- **Iteration** é um WorkDefinition composto que possui um milestone (marco) de pequeno porte.

- **Precondition e Goal** são restrições expressas como expressões booleanas.

A definição de todos estes pacotes de SPEM mostra uma visão geral de como a linguagem é definida e organizada. Agora, procuraremos mostrar como SPEM pode ser utilizada na prática, ou seja, do ponto de vista de seus usuários. De acordo com [48] alguns diagramas básicos da UML podem ser usados para apresentar perspectivas diferentes de um modelo de processo de software.

Em SPEM podem ser utilizados alguns desses diagramas, dentre eles: diagrama de classes, de pacotes, de atividade, de caso de uso, de seqüência e de transição de estados. A linguagem SPEM oferece algumas representações e estereótipos para modelar seus principais elementos em diagramas UML. Um resumo dos principais elementos de SPEM, seus conceitos e suas representações gráficas é mostrado na Tabela A.1.

Tabela A.1 – Estereótipos de SPEM

ESTEREÓTIPO	COMENTÁRIO	NOTAÇÃO
WorkProduct	É uma descrição de algo que contém informação ou é uma entidade física produzida ou usada por atividades do processo. Ex: modelos, planos, documentos, etc.	
WorkDefinition	É um elemento do modelo que descreve a execução, as operações realizadas e as transformações feitas nos "WorkProducts". Corresponde a um conjunto de atividades	
Guidance	É um elemento do modelo que se associa a outros elementos e pode conter descrições adicionais como técnicas, "guidelines", "templates", etc...	

ESTEREÓTIPO	COMENTÁRIO	NOTAÇÃO
Activity	É uma "WorkDefinition" que descreve o que um "ProcessRole" (papel) realiza. Ex: Codificar, Testar.	
ProcessRole	Descreve os papéis, responsabilidades e competências que um determinado indivíduo tem dentro do processo. Exemplos: Testador, Programador, etc.	
Process Component	É um agrupamento coerente de elementos do processo organizados segundo algum ponto de vista como em uma disciplina (Análise e Projeto, teste, implementação, etc.).	

Alguns exemplos de como estes estereótipos podem ser usados já foram mostrados no Capítulo 3 e no Capítulo 5. A descrição de SPEM apresentada neste apêndice fornece uma visão geral sobre a definição da linguagem através do detalhamento dos seus pacotes e estereótipos. A descrição completa de SPEM pode ser encontrada em [48].

Apêndice B – Modelagem de XP em SPEM

Este apêndice contém a modelagem completa de XP que foi descrita parcialmente no Capítulo 4. As figuras a seguir mostram as disciplinas de XP com sua respectiva modelagem através de diagramas de classe estereotipados. Conforme dito anteriormente a criação destas disciplinas se deu através de uma análise interpretativa da literatura de XP, principalmente [28,29]. No final deste apêndice é apresentada novamente a modelagem dinâmica de XP usando diagrama de atividades.

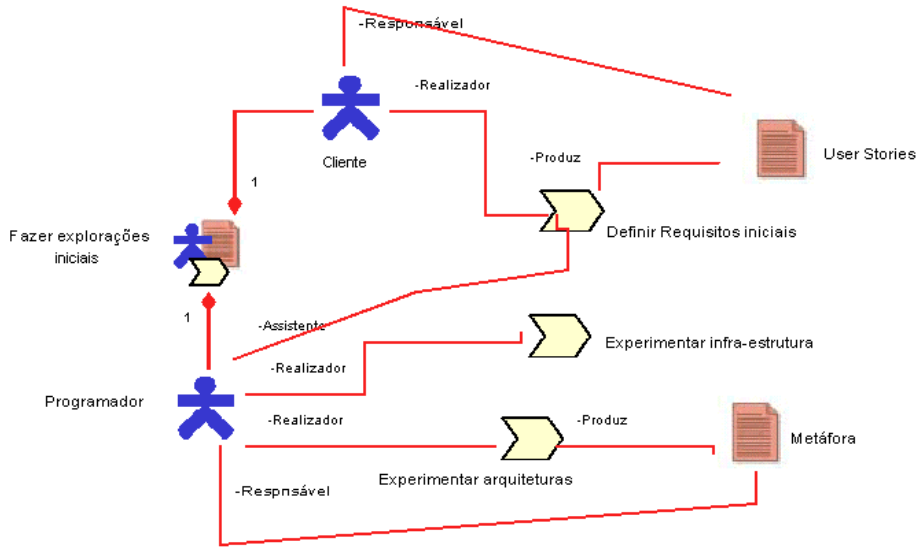


Figura B.1 – Fazer Explorações Iniciais

Fazer Explorações Iniciais: Esta parte do processo XP é onde são feitas as primeiras explorações e investigações sobre o projeto a ser implementado. Os clientes são consultados e trazidos para dentro da equipe para realizar a definição prévia dos requisitos e do escopo do sistema. Os programadores são responsáveis por fazer experimentos com a possível tecnologia e infra-estrutura a ser escolhida para verificar a viabilidade da solução e elaborar uma idéia de arquitetura através de uma metáfora. Algumas estórias iniciais podem ser descritas, mas o importante aqui é levantar as informações necessárias para decidir se o projeto é viável ou não.

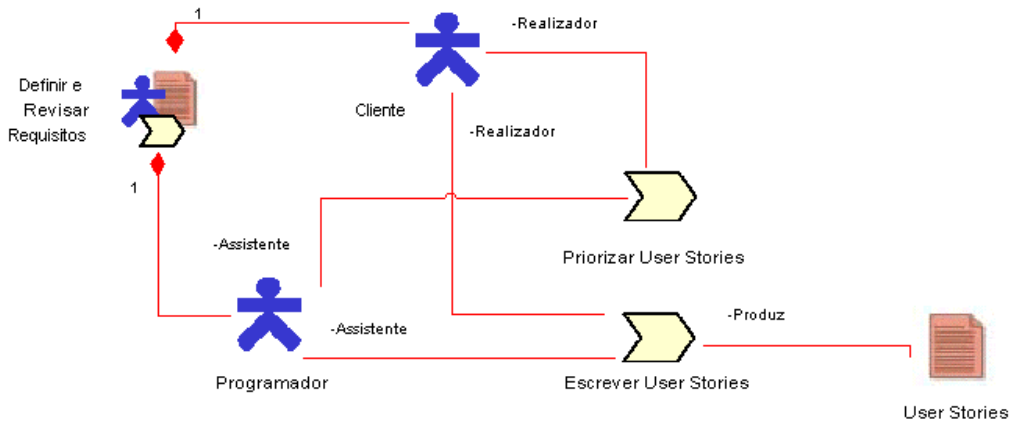


Figura B.2 – Definir e Revisar Requisitos

Definir e Revisar Requisitos: Esta parte do processo XP se preocupa em definir e revisar os requisitos que irão fazer parte de uma versão (release) do sistema. Como XP admite mudanças constantes nos requisitos, então é importante mencionar que esses requisitos podem ser alterados a qualquer instante e que eles também serão revistos na parte de XP que trata dos requisitos da iteração. Em XP, o cliente, ou alguém que o represente, trabalha juntamente com a equipe de desenvolvimento (prática on-site customer) e é responsável por escrever e priorizar as histórias com o auxílio dos programadores. Os clientes escrevem os cartões contendo as descrições de cada estória, as quais são atribuídas uma certa prioridade pelo cliente. Os principais resultados dessa disciplina são os cartões de estória com a descrição de suas funcionalidades e as estimativas de esforço.

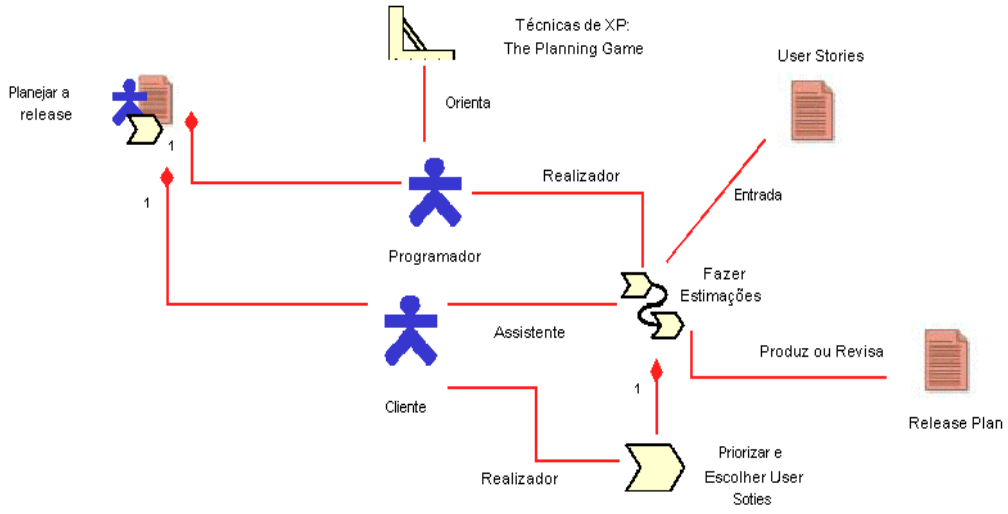


Figura B.3 – Planejar o Release

Planejar o Release: Nesta disciplina é onde se faz o planejamento do que vai ser entregue na próxima versão (release) do sistema. Esse planejamento também pode ser revisado e alterado caso mudem as prioridades do cliente e de acordo com o andamento do projeto. Os programadores utilizam as histórias escritas anteriormente para estimar a sua dificuldade de implementação. A prática "o jogo do planejamento" descreve que um release deve conter as histórias que agregam maior valor para o negócio do cliente, ou sejam, as de maior prioridade para o cliente e os programadores então estimam quantas histórias eles poderão implementar nas diversas iterações do release com base na experiência que possuem sobre a sua velocidade de trabalho e com base nas estimativas de dificuldade de cada história. Com base nisso, eles podem elaborar um plano sobre o que será implementado no release.

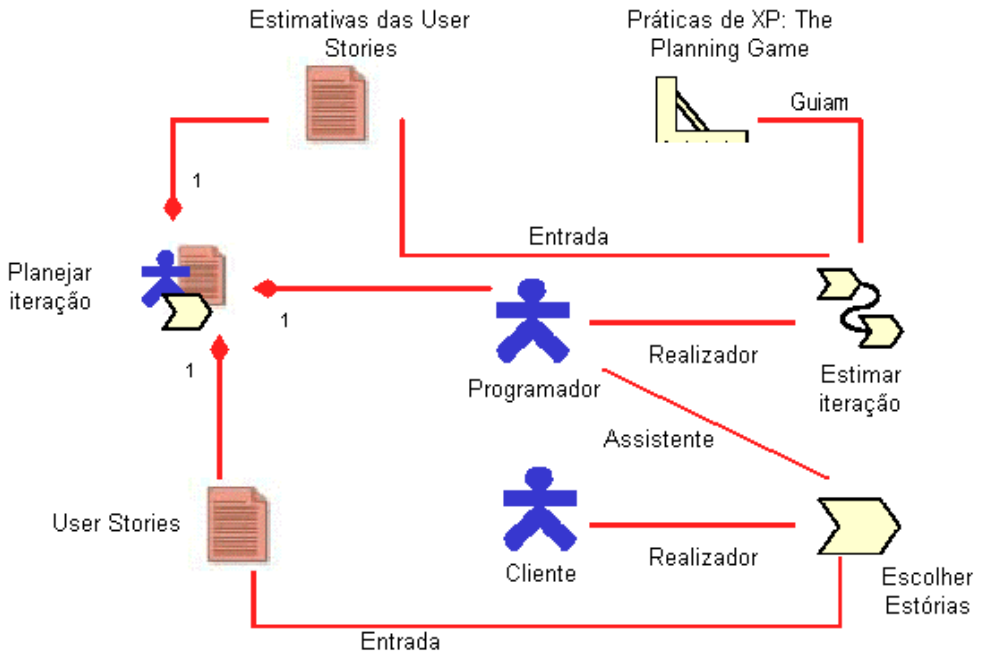


Figura B.4 – Planejar Iteração

Planejar Iteração: Nesta disciplina é onde o planejamento da iteração é feito. Dentro do lançamento de uma versão do sistema (release) podem ocorrer várias iterações em XP. O planejamento da iteração procura refinar as estimativas feitas no planejamento do release e verificar quantas e quais histórias serão implementas na iteração. O programador tenta medir o esforço de uma forma mais precisa e melhorar cada estimativa feita anteriormente no planejamento do release aperfeiçoando as estimativas e mostrando o que pode ser feito dentro de cada iteração ao cliente que tem a responsabilidade de escolher as histórias a serem implementadas. Todo este processo é orientado pela prática de XP jogo do planejamento vista anteriormente. O programador é o responsável pelas estimativas do esforço necessário na implementação de cada história, enquanto o cliente é responsável pela atribuição de prioridades para cada história (coma base em seu valor de negócio) e também por escolher quais devem ser implementadas em cada iteração.

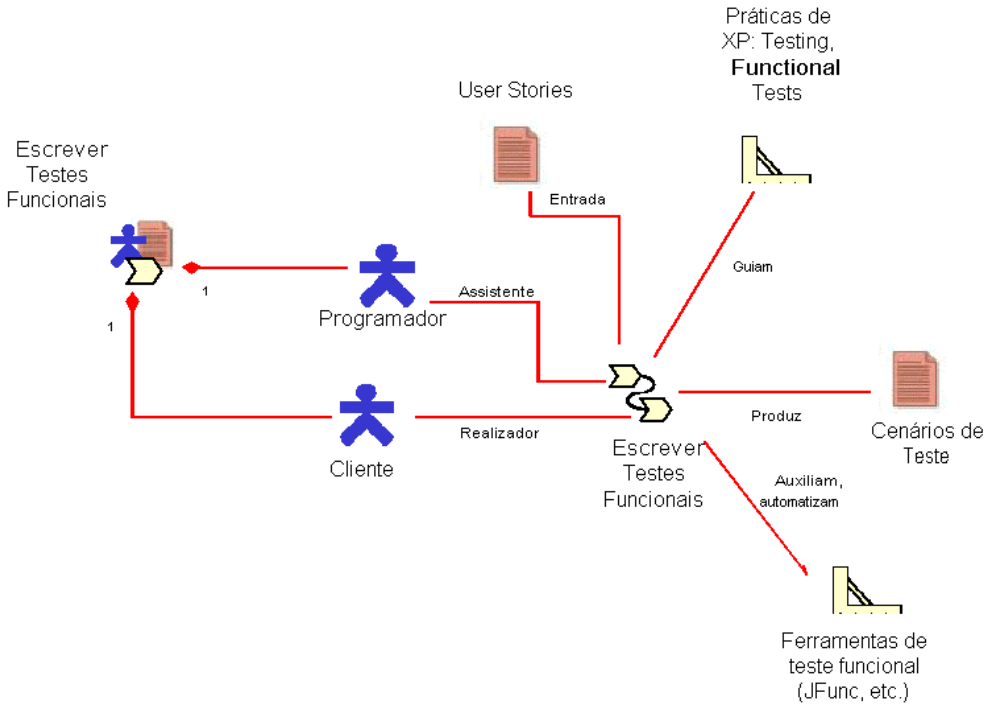


Figura B.5 – Escrever Testes Funcionais

Escrever Testes Funcionais: Nesta parte de XP é onde se realiza a escrita dos testes funcionais. Os clientes escrevem, com o auxílio dos programadores, os cenários de teste que serão usados para validar as funcionalidades do sistema como um todo com base na prática de testes constantes de XP. Esses testes diferem dos testes de unidade que serão descritos posteriormente. O que é produzido aqui são os cenários e o código de teste a serem usados posteriormente, quando as funcionalidades ficarem prontas e forem efetivamente testadas. A decisão de automatizar ou não os testes funcionais pertence à equipe e caso optem pela automação, existem ferramentas disponíveis no mercado, sendo algumas até gratuitas, como por exemplo, JFunc [62], que ajudam a automatizar a tarefa de codificar e executar os casos de teste. Os testes funcionais procuram testar funcionalidades mais complexas (Ex: Venda) que os unitários, cujo objetivo principal é testar os métodos das classes produzidas.

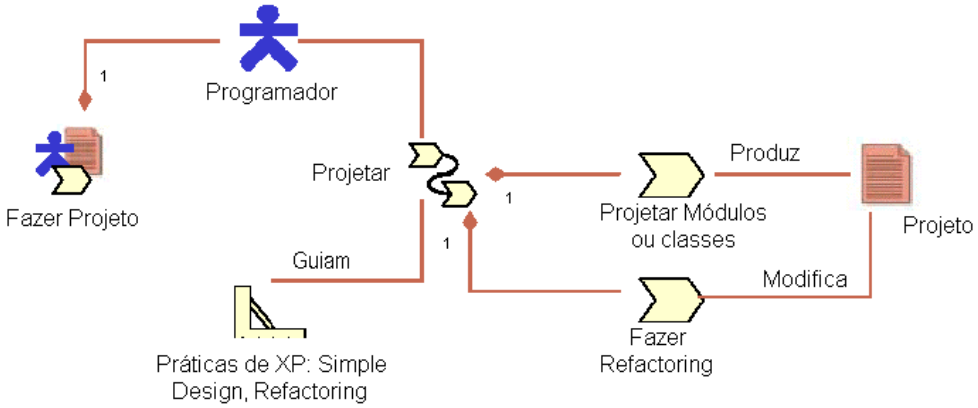


Figura B.6 – Fazer Projeto

Fazer Projeto: Nesta parte de XP é produzido o projeto para as classes que serão implementadas. A idéia aqui é que o projeto seja feito de modo a esclarecer dúvidas entre os programadores da melhor estratégia para implementar as classes do sistema. Não é necessário que se produzam modelos como diagramas de classe UML usando ferramentas CASE, mas os modelos podem ser feitos através de rascunhos em papel ou em um quadro branco e têm o objetivo de esclarecer dúvidas e ser o mais simples possível conforme descrito na prática de XP "Projeto simples". A técnica de refatoramento (refactoring) pode ser aplicada a qualquer momento por qualquer programador que vislumbre uma melhor solução de projeto para o código em desenvolvimento.

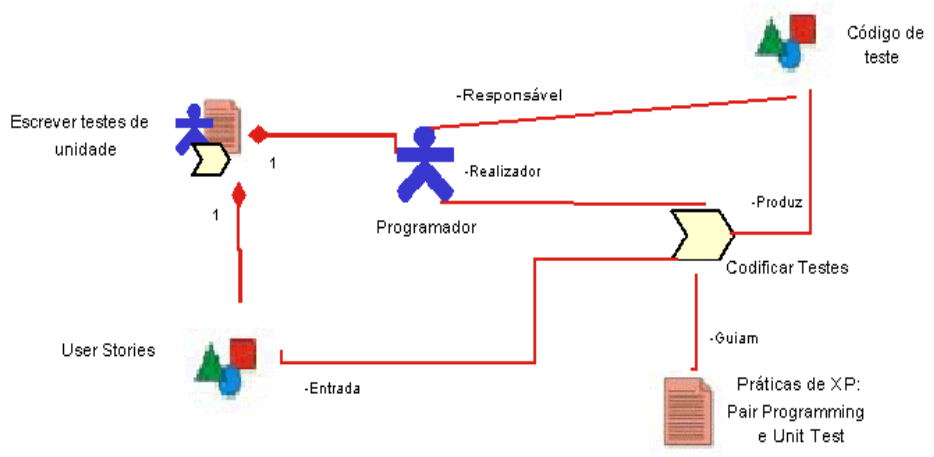


Figura B.7 – Escrever Testes de Unidade

Escrever Testes De Unidade: Nesta parte de XP é onde se produz o código dos testes de unidade. Os testes de unidade correspondem a testes de granularidade mais fina que os testes funcionais. A prática de XP "testes unitários" orienta que se deve testar tudo que possa dar errado, o que implica em escrever testes para as principais operações das classes. Os programadores escrevem em pares (programação em pares) o código de testes que será depois validado quando as classes forem implementadas. O código de teste nada mais é que o código comum de uma classe que possui alguns comandos especiais para verificações de valores, como se fossem asserções. Esse código pode ser produzido com o auxílio de ferramentas de teste (JUnit) que possuem uma interface gráfica que permite executar todos os testes produzidos e apontar os erros caso ocorram.

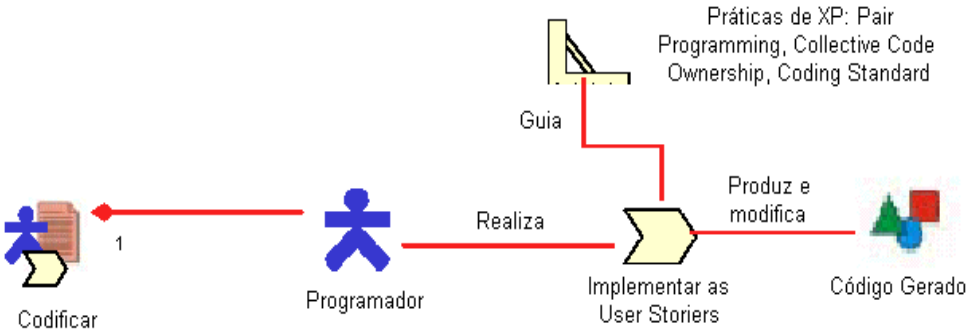


Figura B.8 – Codificar

Codificar: Esta disciplina forma o núcleo de XP que é a programação. Os programadores se organizam em pares (programação em pares) e implementam as histórias descritas anteriormente. Os programadores devem seguir um padrão de codificação e não possuem domínio absoluto sobre nenhuma parte do código que é de posse coletiva e que pode ser melhorado e alterado por qualquer programador da equipe. As duplas devem ser constantemente trocadas para implementar funcionalidades diferentes. Além disso, o papel de cada programador na dupla também deve variar. Isso possibilita que todos os membros da equipe tenham uma visão geral do sistema. A implementação das classes de negócio do sistema é feita em conjunto com a implementação dos casos de teste pela dupla de programadores. Kent Beck sugere em [28] que durante a construção de um sistema em XP sempre existam ciclos diários das

atividades de programação, escrita de testes, refatoramento, realização de testes e integração.

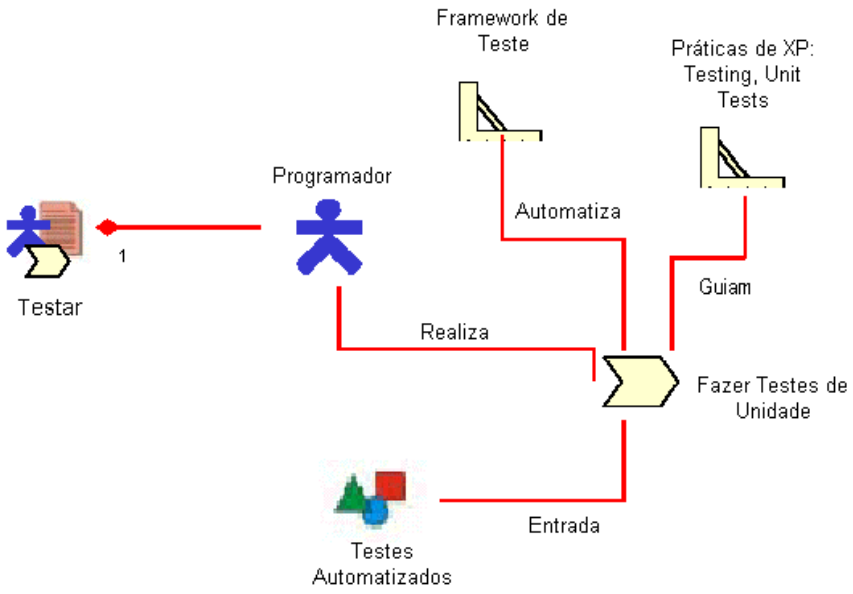


Figura B.9 – Testar

Testar: Nesta parte de XP executam-se os testes de unidade. Esses testes já foram codificados anteriormente e aqui eles serão executados pelos programadores com o auxilio de um Framework de testes como JUnit, por exemplo, que automatiza a verificação dos testes. Caso os testes não passem em 100% corretos, os erros devem ser encontrados e corrigidos. O objetivo desta disciplina de XP é manter a qualidade do código que é produzido e forçar que novas funcionalidades implementadas não introduzam erros. Os testes são sempre armazenados e toda vez que as novas funcionalidades são implementadas, todos os casos de teste, desde o mais antigo até o mais recente, são executados e devem estar 100% corretos.

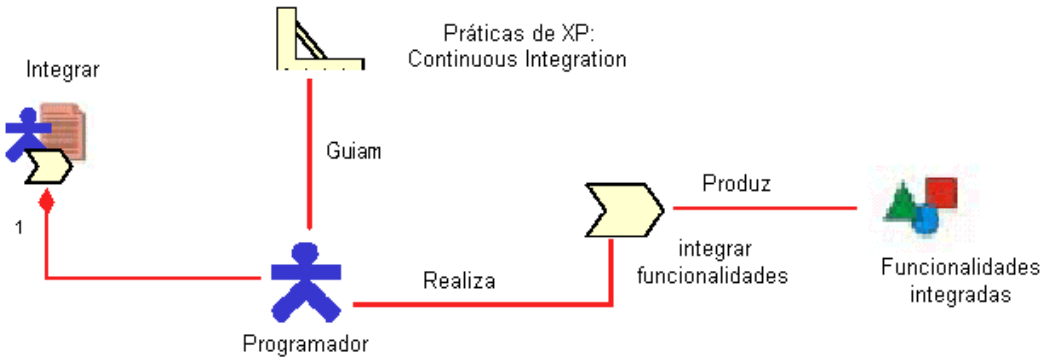


Figura B.10 – Integrar

Integrar: Este diagrama mostra a integração em XP. As classes construídas e testadas através de testes de unidade devem ser integradas entre si para compor os módulos do sistema. A integração pode ocorrer de forma muito freqüente em um projeto XP, podendo ocorrer várias vezes por dia à medida que as funcionalidades vão sendo implementadas e testadas. A prática de XP "Integração Contínua" orienta como isso deve ser feito. Ao integrar as funcionalidades é importante verificar se os testes respeitam o que foi especificado nos cenários de teste que podem ou não ter sido automatizados. Além disso, cada vez que se integrar uma funcionalidade, todos os casos de teste unitários devem ser executados, e caso exista algum erro, este deve ser corrigido de imediato.

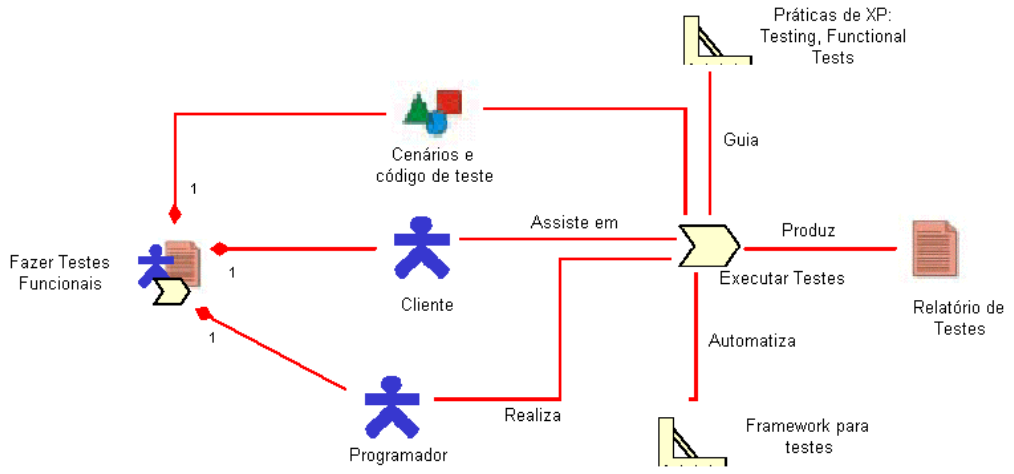


Figura B.11 – Fazer Testes Funcionais

Fazer Testes Funcionais: Este diagrama mostra o que é realizado nos testes funcionais de XP. O programador é quem realiza os testes funcionais com o auxílio do

cliente. Os cenários e o código de teste produzido anteriormente são utilizados como entrada para a execução dos testes que podem ser automatizados por ferramentas de teste (Frameworks de teste como JUnit e JFunc) e um relatório de testes pode ser produzido reportando os erros encontrados. O objetivo destes testes é verificar se as funcionalidades estão de acordo com as necessidades e expectativas do cliente. O que se testa neste caso são funcionalidades como um todo (Ex: Vendas, Estoque, etc.) e não apenas métodos. Os cenários de teste escritos servem para guiar a realização dos testes que podem ou não ser automatizados.

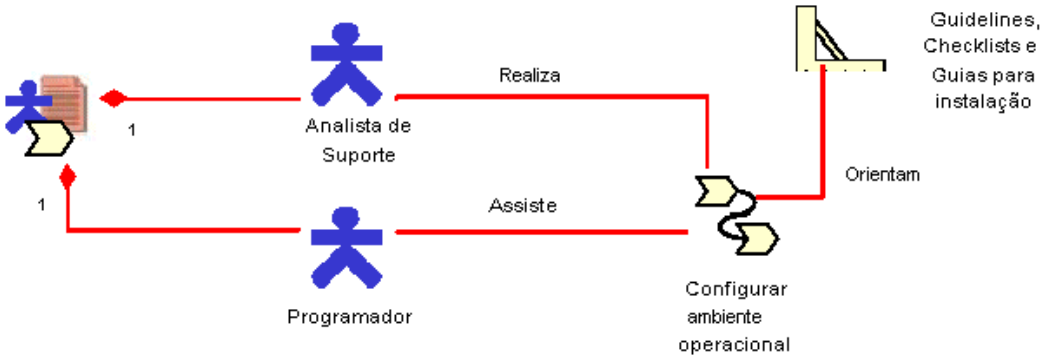


Figura B.12 – Colocar em Produção

Colocar em Produção: Após várias iterações uma versão (release) do sistema é produzida e deve ser posta em produção no ambiente operacional. O pessoal de suporte realiza a configuração da infra-estrutura de hardware e software na qual o sistema irá operar no cliente e coloca a versão atual do sistema em execução. Essa atividade pode ser auxiliada por documentos e guias produzidas como um manual ou listas de verificação (checklists). O local efetivo onde o release será colocado em operação pode ser: no próprio ambiente do cliente, que pode optar por utilizar o sistema como experiência ou colocá-lo já em funcionamento; ou em um ambiente que simule o ambiente do cliente para verificar questões como eficiência e desempenho.

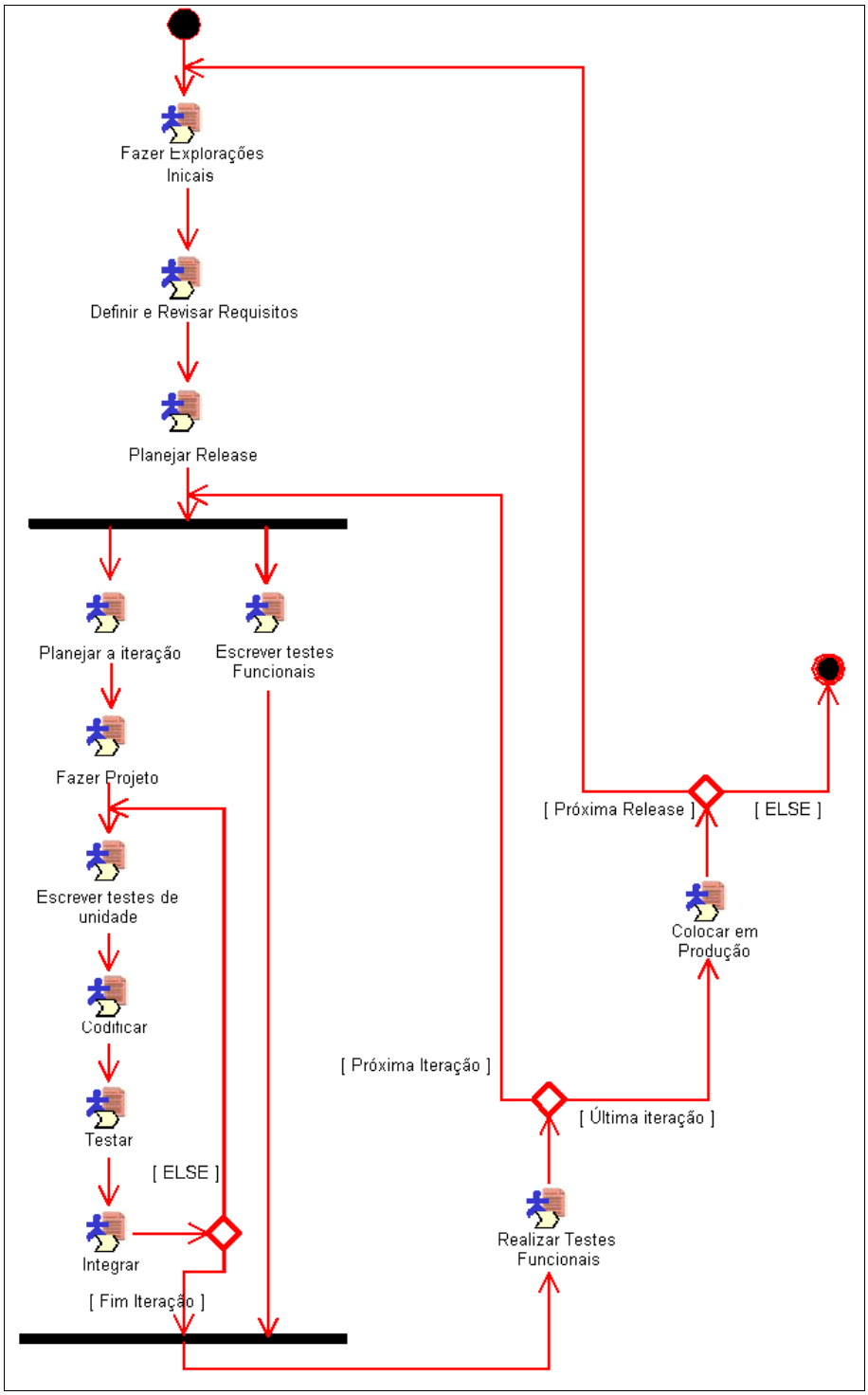


Figura B.13 – Modelagem dinâmica de XP

Apêndice C – Documentos do Experimento

Este apêndice apresenta alguns documentos e modelos que foram usados no experimento. Primeiramente será mostrado o questionário que foi distribuído a todos os alunos que participaram do experimento. O objetivo deste questionário foi o de obter informações sobre a qualidade da definição e modelagem dos processos usados (XP e XWebProcess) bem como compará-los em relação ao desenvolvimento Web. O questionário é mostrado a seguir.

QUESTIONÁRIO

1) Quais as principais dificuldades e vantagens de trabalhar com o seu processo (XP/XWebProcess) em um projeto Web na sua opinião? Justifique a sua resposta.

Quanto à modelagem feita de XP/XWebProcess com SPEM responda os itens de 2 a 4 abaixo.

2) Facilidade de compreensão: Até que ponto o processo está explicitamente definido e com que facilidade se pode compreender a definição (modelagem) do processo?

( ) Mal Definido: As atividades do processo são mal definidas. Não se tem idéia da responsabilidade de cada papel e nem qual a seqüência de atividades que deve ser seguida ao longo do tempo.

( ) Razoavelmente Definido: As atividades do processo são bem definidas e explicadas. No entanto, não se tem uma idéia muito detalhada da responsabilidade de cada papel e apenas uma idéia razoável da seqüência das atividades ao longo do projeto.

( ) Bem definido: As atividades do processo estão bem definidas e explicadas. Tem se uma idéia clara pela definição do processo das responsabilidades de cada papel e da seqüência que as atividades seguem desde o início até o final do projeto.

Justifique a sua resposta:

3) Visibilidade: As atividades do processo culminam em resultados nítidos, de modo que o progresso do projeto seja externamente visível? Avalie o processo quanto às entradas e saídas de cada atividade (artefatos gerados, atualizados e descartados)

( ) Fracamente Visível: Não se tem uma idéia clara dos artefatos gerados ou atualizados em cada atividade e nem quem é responsável por ela. Também não se sabe quais artefatos podem ou não ser descartáveis

( ) Visível: As atividades do processo definem de forma clara os artefatos de entrada e saída e os responsáveis por ele, porém não se sabe a respeito da sua importância (descartável ou não)

( ) Bem visível: As atividades do processo definem de forma clara os artefatos de entrada e saída e os responsáveis por ele e sabe-se quais artefatos são essenciais ou não.

Justifique a sua resposta:

4) Facilidade de manutenção: O processo pode evoluir para refletir os requisitos mutáveis da organização ou melhorias de processo identificadas? Você acredita que seria fácil fazer alterações no processo para englobar novas atividades, papéis e outros elementos?

( ) Difícil de alterar: Devido a sua definição inadequada fica difícil compreender o processo e por conseqüência alterá-lo para atender às necessidades da organização ou de projetos em particular

( ) Fácil de alterar: Devido ao processo ser bem definido fica mais fácil fazer adaptações nele para atender a necessidades específicas de projeto e da organização.

Justifique a sua resposta:

5) Faça uma comparação entre XP e XWebProcess. Você acredita que os elementos inseridos e modificados no processo XWebProcess são necessários (Faça uma análise rápida sobre cada uma das 6 disciplinas em destaque na Figura 5.2)? Que benefícios ou desvantagens eles podem trazer? Qual dos dois processos é o mais adequado para o desenvolvimento Web?

Justifique a sua resposta:

Além deste questionário outros documentos importantes utilizados neste experimento foram: o modelo para escrita das estórias (User Stories), o modelo para testes funcionais e o modelo de projeto de navegação e interface (utilizado apenas pelas equipes de XWebProcess).

MODELO PARA ESTÓRIAS

Data: 10/12/2003
Tipo de Atividade: Nova: _X_ Correção: ___ Melhoria: ___ Teste F.___

Número da Estória: 4
Prioridade: Usuário: _5_ Técnica: _5_
Nome da estória / tarefa: Cadastro de Cliente___
Estimativa Técnica: __7 dias__

DESCRIÇÃO DA ESTÓRIA / TAREFA:

A funcionalidade de cadastro de vendas será efetuada pelo cliente. Após efetuar o login, o cliente poderá consultar todos os livros da livraria. Através desta listagem com os dados dos livros, o cliente irá adicionar novos itens ao seu carrinho de compras. O carrinho de compras terá os títulos selecionados com suas respectivas quantidades, preços, e por fim, o preço total.

No ato de fechamento da compra, o sistema pedirá uma confirmação, e, caso seja positiva, irá gerar um código que será exibido para o cliente fazer consultas do andamento do seu pedido. Será também armazenada a data de realização da venda.

Os dados requisitados serão: conjunto de livros com respectivas quantidades, data da venda e cliente que estará efetuando a compra.

NOTAS:

A chave primária para a venda será o código gerado pelo sistema.

MODELO PARA TESTES FUNCIONAIS

Data: 2/12/2003
Tipo de Atividade: Nova: Correção: __ Melhoria: ___ Teste F. X

Número da Estória: 1 (Inserir Funcionário)
Prioridade: Usuário: 5 Técnica: 4
Nome da estória / tarefa: Caso de Teste (Inserir Funcionário)
Estimativa Técnica: 7 dias

DESCRIÇÃO DA ESTÓRIA / TAREFA:

1. Fluxo Normal

Entradas

- O sistema deve estar no ar.
- No menu onde estão listados os funcionários, o usuário escolhe a opção *novo funcionário*.
- Na próxima página o usuário coloca diversas informações referentes ao funcionário (Nome, CPF, Telefone, Data, Login, senha).

Resultados

- Ao clicar na opção *efetuar inserção a* página é automaticamente redirecionada para a lista de funcionários com o funcionário já inserido.

Condições

- Para executar a inserção de funcionários, o funcionário(e somente funcionários podem inserir funcionários) deve ter logado no sistema.

2. Fluxo Alternativo 1

Entradas

- O sistema deve estar no ar
- No menu onde estão listados os funcionários, o usuário escolhe a opção *novo funcionário.*
- Na próxima página o usuário coloca um login já existente no sistema.

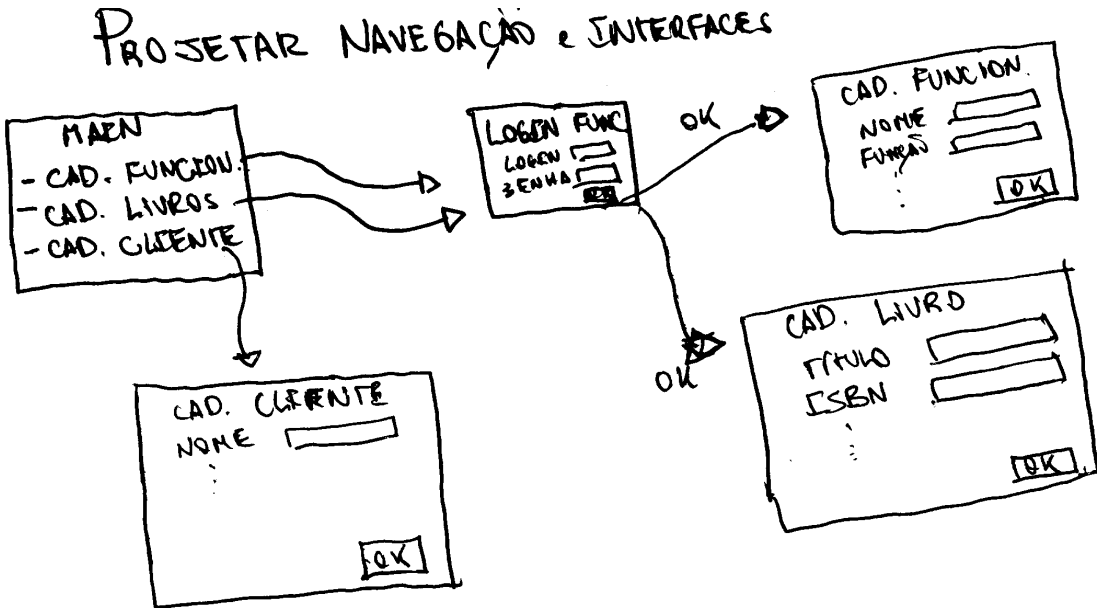
Resultados

- O sistema apresenta na tela um pedido para o usuário digitar um novo login.

Condições

- Para executar a inserção de funcionários, o funcionário(e somente funcionários podem inserir funcionários) deve ter logado no sistema.

MODELO PARA PROJETO DE NAVEGAÇÃO E INTERFACE



Este modelo já havia sido sugerido na Seção 2.4.3 como uma forma de elaborar um projeto da interface gráfica Web de forma rápida e que serve para dar uma idéia da estrutura de cada página além da navegação entre as páginas. O modelo acima foi gerado através de um rascunho em papel que foi posteriormente escaneado. Os grupos que usaram XWebProcess tiveram a obrigação de gerar este artefato como parte da disciplina de projetar navegação e interface Web.

Dissertação de Mestrado apresentada por **Américo Tadeu Falcone Sampaio** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título , **"XWEBPROCESS: Um Processo Ágil para o Desenvolvimento de Aplicações Web"**, orientada pelo **Prof. Alexandre Marcos Lins de Vasconcelos** e aprovada pela Banca Examinadora formada pelos professores.

Prof. Hermano Perrelli de Moura
Centro de Informática / UFPE

Prof. Jones Oliveira de Albuquerque
Departamento de Física e Matemática / UFRPE

Prof. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 23 de março de 2004.

Prof. JAELSON FREIRE BRELAZ DE CASTRO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.