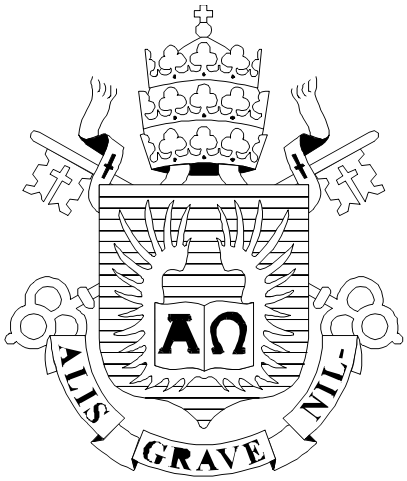

PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 23/09

Processos para Desenvolvimento de Aplicações Web

Mark Douglas de Azevedo Jacyntho

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 23/09 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Julho, 2008

Processos para Desenvolvimento de Aplicações Web

Mark Douglas de Azevedo Jacyntho

mjacyntho@inf.puc-rio.br

Abstract. Web application development presents significant differences from conventional application development. The spectrum varies from technical to organizational differences. Technical means the specific architectures and technologies employed and the impacts involved. Organizational is related to the strategic use of these applications aiming at improving the business. Among other issues, uncertainty, volatility and high competitiveness are innate characteristics which must be carefully addressed. Therefore, the nature of web engineering suggests the need of specialized software processes that cover, in a systematic way, the complete life cycle of hypermedia web applications, in contrast to adopting ad-hoc approaches to comply with the constraints imposed by this application domain. This essay presents an abridge discussion concerned with the impact of these differences on development process for web applications, analyzing some proposals, indentifying the requirements and challenges. This work intends to be an initial step towards the definition of a process suitable to web development, in dimensions such as requirements gathering, user interface, testing and navigation design.

Keywords: web process, web engineering, web development, software process, software engineering, software development.

Resumo. Desenvolvimento de aplicações web apresenta diferenças significativas com relação ao desenvolvimento de aplicações convencionais. O espectro varia desde diferenças técnicas até organizacionais. Diferenças técnicas significam as arquiteturas e tecnologias específicas empregadas e os impactos envolvidos. Já as organizacionais são relacionadas ao uso estratégico destas aplicações visando melhorar o negócio. Dentre outros fatores, incerteza, volatilidade e alta competitividade são características inerentes que precisam ser consideradas. Por conseguinte, a natureza da engenharia para web sugere a necessidade de processos de software especializados que atendam, de forma sistemática, o ciclo de vida completo de aplicações web hipermídia, em contraste com a adoção de abordagens ad-hoc para lidar com as restrições impostas por este domínio de aplicações. Esta monografia apresenta uma breve discussão sobre o impacto destas diferenças em processos de desenvolvimento para aplicações web, analisando algumas propostas, identificando os requisitos e desafios. Este trabalho pretende ser um passo inicial rumo à definição de um processo apropriado para o desenvolvimento web, em dimensões como levantamento de requisitos, interface com o usuário, testes e projeto de navegação.

Palavras-chave: processo web, engenharia para web, desenvolvimento web, processo de software, engenharia de software, desenvolvimento de software.

---

* Trabalho patrocinado pelo Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil (e CNPq, processo: 142192/2007-4).

Responsável por publicações:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Sumário

Índice de Figuras

1 Introdução

Vários processos de software têm sido propostos ao longo dos anos. Essencialmente todos tentam definir um *roadmap* que guie o desenvolvimento, identificando quem está fazendo o quê, onde, por que, como e quando. Um processo de software é definido com um conjunto de atividades interdependentes que visam desenvolver, manter e gerenciar sistemas de software. Estas atividades podem ser compostas de outras atividades e são executadas por atores que desempenham um papel no processo (programador, gerente, cliente, etc.). Como resultado das atividades, são produzidos artefatos (código, documentação, modelos) que servem de entrada para outras atividades para produzir novos artefatos. Sem um processo de software, o risco de falha do projeto se torna muito alto, em especial para as aplicações web modernas cuja complexidade não pára de crescer.

Com o passar dos anos, as aplicações web evoluíram rapidamente de simples web sites cujo propósito era apenas navegação sobre a informação para verdadeiros sistemas de informação altamente complexos, repletos de dados e transações, voltados para a implementação de processos de negócio intra- e inter-organização. Diante deste quadro, a necessidade de um processo de software sistemático que ajude a gerenciar o ciclo de vida de tais aplicações surge naturalmente.

Poder-se-ia argumentar que sistemas web não são diferentes, sob o ponto de vista de processo de software, de sistemas de software convencionais. Não obstante, existe um crescente reconhecimento que sistemas web possuem características particulares que não são apropriadamente consideradas pelos processos de software tradicionais. O desenvolvimento de aplicações web é realizado por equipes multidisciplinares, com diferentes habilidades, em prazos curtíssimos ditados pela voraz concorrência e em um contexto extremamente volúvel, marcado por incertezas. Para completar, sistemas web são tecnologicamente muito abstrusos, reunindo padrões, protocolos e tecnologias diversas na definição de uma arquitetura que encapsule em um *front-end* amigável um *back-end* que pode ser por demasiado complexo e heterogêneo.

Em muitos casos, as equipes de desenvolvimento, acometidas pelas severas restrições de tempo, adotam soluções *ad-hoc* para construir tais aplicações. Neste cenário, o sucesso do projeto depende muito da habilidade e conhecimento dos membros da equipe, com os usuais efeitos colaterais negativos em flexibilidade, qualidade e robustez da aplicação [Sampaio et al, 2004a].

Como é de se esperar, paralelamente às diferenças, também co-existem pontos em comum entre aplicações web e convencionais. Portanto, um caminho interessante seria delinear as peculiaridades deste domínio de aplicação e adaptar ou enriquecer processos de software já existentes de forma a contemplar mais claramente as necessidades específicas da web.

Sob a ótica deste trabalho, é importante ressaltar a diferença entre aplicação na web e aplicação web. Aplicação na web é qualquer tipo de aplicação que utiliza a web como ambiente de execução. Um simples repositório de arquivos ou uma aplicação com estilo tradicional *desktop*, composta apenas por buscas e formulários, são exemplos de aplicação na web. Já aplicações web são aquelas que, necessariamente, exploram o paradigma hipermídia. Em outras palavras, no contexto deste trabalho, somente são consideradas aplicações web aquelas que possuem uma estrutura navegacional bem defini-

da, fazendo jus ao elemento fundamental da web que é a noção de *hiperlink*. O grande desafio das aplicações web modernas é integrar, elegantemente, dois paradigmas capitais: hipermídia e transação. Sendo assim, este trabalho procurar evidenciar os requisitos que um processo de desenvolvimento de software precisa atender de modo a ser, efetivamente útil, nesta integração.

Esta monografia considera as diferenças entre aplicações web e convencionais, destacando as implicações na definição de um processo de software para web. Para tal, é definida uma lista de requisitos que um processo web deve atender e, com base nesta, são analisadas duas propostas existentes.

1.1 Processos Orientados a Plano versus Processos Ágeis

Mais recentemente os processos de software passaram a ser classificados em duas categorias: orientados a plano e ágeis.

Processos orientados a planos são processos mais rigorosos, preditivos por natureza, onde existe um plano que procura antever problemas e as respectivas soluções. Ao longo do desenvolvimento todas as decisões de projeto são documentadas e controladas, antes de serem implementadas. O foco é no processo, ou seja, institucionalizar um processo de software e segui-lo a risca. Este processo deve ser continuamente aprimorado. São exemplos: CMMI - *Capability Maturity Model Integration* [Chrissis et al, 2003], TSP - *Team Software Process* [Humphrey, 2000], PSP - *Personal Software Process* [Humphrey, 1995], SPICE - *Software Process Improvement and Capability dEtermination* [SPICE].

Por outro lado, processos ágeis têm o foco no produto em si, ou seja, o mais importante é entregar software em detrimento de documentação. São reativos por natureza, onde atitudes são tomadas sob demanda e é desenvolvido somente o necessário e quando necessário. Trata-se de um desenvolvimento iterativo e incremental onde, constantemente, são entregues novos *releases* do produto ao cliente. As idéias podem ser encontradas no "*Agile Manifesto*" [Agile Manifesto]. Os elementos centrais do manifesto são:

- Pessoas e interações são mais importantes do que processos e ferramentas;
- Software executável é mais importante do que documentação;
- Colaboração do cliente é mais importante do que negociação por contrato;
- Reação a mudanças é mais importante do que plano pré-definido.

Processos rigorosos e processos ágeis, ambos têm suas vantagens e desvantagens. Não existe processo correto ou incorreto, existe processo adequado e inadequado, ou seja, para cada tipo de projeto é necessário encontrar um ponto de equilíbrio entre as duas abordagens e definir um processo híbrido que traga benefícios reais [Bohem e Turner, 2004].

Independentemente de quão rigoroso ou ágil seja o processo, o fato é que temos que levar em consideração como contemplar os requisitos presentes no desenvolvimento web, discutidos ao longo deste trabalho. No entanto, o caráter altamente mutável das aplicações web nos leva a crer que uma abordagem mais ágil venha mais ao encontro deste domínio de aplicações do que um processo mais rigoroso.

1.2 Organização da Monografia

A secção 2 explora as diferenças entre sistemas web e sistemas convencionais. Em seguida, na secção 3, são enumerados os requisitos que devem ser levados em consideração quando da elaboração de um processo de desenvolvimento para web. Para tornar a discussão mais concreta, a secção 4 apresenta duas propostas de processo web existentes. Seguindo na mesma linha, na secção 5, as propostas apresentadas são analisadas de acordo com os requisitos descritos anteriormente. Encerrando, a secção 6 delineia as considerações finais.

2 Diferenças entre Aplicações Web e Aplicações Convencionais

Existe um conjunto de diferenças que justificam a necessidade de uma atenção especial ao desenvolvimento de aplicações web. Algumas características são únicas de sistemas web, outras também estão presentes nos sistemas convencionais, mas são mais pronunciadas na web. Com base em [Lowe e Henderson-Sellers, 2001] e [Kappel et al, 2004], estão secção destaca tais diferenças que servem como base para a definirmos os requisitos necessários para a definição de um processo de desenvolvimento para web. A distinção é feita primeiramente sob um enfoque técnico e, em seguido, questões organizacionais são discutidas.

2.1 Diferenças Técnicas

Claramente existem diferenças técnicas entre sistemas web e convencionais. As mais significantes são:

- Diferenças relativas à aplicação:
    - Conteúdo: a web é essencialmente um meio de informação. Além da funcionalidade, uma aplicação web é orientada a conteúdo. Conteúdo compreende dados estruturados (banco de dados, por exemplo) e não estruturados (arquivos textos, vídeos, etc.). Além disso, o conteúdo é dinâmico, precisa ser continuamente atualizado e de qualidade em termos de consistência e confiabilidade. Isto implica em um efetivo projeto de informação, bem como gerenciamento de conteúdo apropriado.
    - Hipertexto: na web o paradigma fundamental para estruturar a informação é noção de hipertexto, onde os elementos básicos são: nós, elos (*links*) e âncoras que ativam estes elos. Os nós exibem informações e os elos nos permitem navegar entre os nós. Isto requer um projeto cuidadoso e estratégico de navegação que preserve a qualidade de acesso e evite desorientação e sobrecarga cognitiva.
    - Apresentação: o "*look and feel*" da aplicação web é um fator de qualidade essencial uma vez que usuários podem facilmente abandonar o site e ir para outro concorrente. Não existe um manual de usuário, portanto a interface tem que ser auto-explicativa, intuitiva e consistente com o estilo de interação. Além disso, a aparência visual está sujeita a modismo, tendências e novas características técnicas que surgem todos os dias.

- Requisitos não-funcionais de qualidade: requisitos de qualidade como disponibilidade 24/7, performance, usabilidade, escalabilidade, robustez e segurança se tornam ainda mais críticos quando expostos externamente ao público.

- Diferenças relativas ao uso:
  - Ubiqüidade: à medida que a necessidade de ubiqüidade (ou seja, a possibilidade de acessar a aplicação por diferentes tipos de dispositivos, em diferentes contextos de uso) das aplicações cresce, torna-se necessário projetar diferentes visões da aplicação, uma para cada contexto de uso.
  - Infra-estrutura tecnológica imprevisível: juntamente com o crescente caráter ubíquo, surge uma enorme variedade de dispositivos de usuário final, com diferentes capacidades de hardware e software, diferentes tamanhos e versões de navegadores. Conexões de rede diferem com respeito à largura de banda, confiabilidade, estabilidade e disponibilidade, afetando a questão de *QoS* (qualidade de serviço). Por fim, usuários configuram livremente seus navegadores, podendo desabilitar importantes funcionalidades.
- Diferenças relativas ao desenvolvimento:
  - Ambiente de desenvolvimento: a infra-estrutura técnica usada para desenvolver sistemas web é caracterizada por exacerbada volatilidade e heterogeneidade. As aplicações são freqüentemente construídas a partir de componente COTS (*commercial off-the-shelf*) que são adaptados e integrados, particularmente para as camadas internas de *middleware*. Dentre estes componentes se destacam servidores de aplicação e frameworks de aplicação e persistência. Isto aumenta a importância de criar soluções flexíveis que possam migrar para novas tecnologias com pouco esforço. Outra conseqüência é o domínio restrito destas tecnologias por parte da equipe, aumentando o risco do projeto e demandando um constante aprimoramento do conhecimento.
  - Integração com sistemas legados: aplicações web costumeiramente precisam se integrar com sistema legados. São sistemas com interfaces antigas e inapropriadas que, portanto, precisam ser encapsuladas e adaptadas (*wrapping*). Este encapsulamento é trabalhoso e existe o risco de não ser feito de forma correta e completa. Sistemas legados raramente são documentados e freqüentemente são modificados sem notificação, afetando a execução do sistema web. Além disso, as tecnologias para encapsulamento são várias e estão sempre mudando (por exemplo, encapsulamento via *web services*).

2.2 Diferenças Organizacionais

Sob o ponto de vista organizacional, as diferenças mais proeminentes são:

- Diferenças relativas ao desenvolvimento:
    - Incerteza do cliente: devido à alta dinamicidade da web, este domínio de aplicações não é bem compreendido pelos interessados (também chamados de *stakeholders)*. Freqüentemente o cliente tem problemas em articular suas necessidades, bem como em compreender se um determinado design satisfaz suas expectativas. Além disso, é muito comum a existência de projetos orientados por uma idéia visionária ao invés de uma necessidade bem definida. Isto aumenta a necessidade de desenvolvimento incremental baseado em protótipos.
    - Alta volatilidade dos requisitos de negócio: fortemente relacionada à questão anterior está a falta de clareza, por parte do cliente, sobre os reais impactos que a aplicação web traz para o negócio. Sendo assim, não é surpresa esperar que o escopo e foco do projeto mudem consideravelmente ao longo do desenvolvimento. À medida que a empresa vai aumentando sua participação na web, o próprio modelo de negócio da empresa pode mudar diante de novas oportunidades outrora desconhecidas.
    - Ciclos de desenvolvimento muito curtos: projetos web, em geral, têm prazos mais curtos do que projetos convencionais, comumente de um a três meses.
    - Alta competitividade: a concorrência na web é muito mais intensificada pela facilidade do usuário final visitar vários sites em um curto espaço de tempo. Portanto, a aplicação precisa estar sempre atualizada e atraente de acordo com as tendências atuais.
    - Equipes multidisciplinares: o desenvolvimento de uma aplicação web é um esforço multidisciplinar, envolvendo profissionais das mais diversas áreas, com habilidades bem díspares. São designers gráficos, autores, engenheiros de software, programadores, publicitários, consultores de negócio, entre outros. Em geral, há muito jovens inexperientes inclinados a empregar tecnologias novas de forma ad-hoc.
    - Evolução e manutenção de granularidade fina: aplicações web estão sujeitas a mudanças diárias e permanente evolução. Tipicamente temos um processo contínuo de atualização de conteúdo, mudanças editoriais, ajustes de interface, etc. Sem falar nas mudanças tecnológicas mais radicais.
- Diferenças relativas ao uso:
    - Diversidade de tipos de usuário: na web os usuários variam em idade, conhecimento sócio-cultural, objetivos, intenções, habilidades e capacidades. Esta heterogeneidade, diferentes perfis de usuário, precisa ser considerada, dado que na web os usuários são inteiramente livres para escolherem as aplicações que lhe forem mais convenientes.

- Sazonalidade (ou características temporárias): aplicações possuem requisitos que são sazonais ou temporários, ou seja, informações e/ou funcionalidades que são oferecidas, estrategicamente, por uma faixa de tempo, que pode se repetir ou não. Um exemplo são os contextos navegacionais relacionados ao período natalino.

Analisando o que foi exposto nesta secção, percebe-se que um processo de desenvolvimento voltado para aplicações web é caracterizado por freqüentes mudanças e ajustes, que são necessárias devido à rápida evolução tecnológica, ao surgimento de novas tendências, à volatilidade dos requisitos e aos cronogramas apertados. Um processo para web precisa ser altamente iterativo, flexível e orientado a protótipos.

3 Requisitos de um Processo de Desenvolvimento Web

Sob a luz das diferenças descritas anteriormente e do trabalho exposto em [McDonald e Welland, 2004], foram identificados dezessete requisitos que um processo de desenvolvimento de aplicações web deve ponderar. Como era de se esperar, estes requisitos se confundem com os requisitos que um método de especificação de aplicações web (por exemplo, OOHDM [Rossi, 1996]) deve contemplar, reforçando a importância do emprego de um método desta natureza ao longo do processo. Sendo assim, um processo web deve dar suporte para:

1. Autoria (ou engenharia) de conteúdo;
2. Autoria (ou engenharia) de navegação sobre o conteúdo;
3. Autoria (ou engenharia) de apresentação (interface);
4. Ubiqüidade;
5. Definição de arquitetura multicamadas
   5.1. Escolha, adaptação e utilização de frameworks e componentes;
   5.2. Integração com sistemas legados;
   5.3. Distribuição das camadas entre servidores;
   5.4. Definição do que roda no navegador e do que roda no servidor;
6. Ciclos de vida de desenvolvimento curtos;
7. Equipes multidisciplinares;
8. Desenvolvimento concorrente de pequenas equipes em tarefas interdependentes;
9. Pesquisa de mercado;
10. Incerteza do cliente e volatilidade dos requisitos;
11. Reengenharia de modelos de negócio a partir de modelos da aplicação;
12. Análise de negócio e avaliação junto ao usuário final;
13. Diferentes perfis de usuário (personalização);
14. Requisitos explícitos (funcionais e não-funcionais);
15. Testes rigorosos com relação aos requisitos;
    15.1. Teste de performance, escalabilidade e resiliência;

16. Autoria (engenharia) de Segurança
    16.1. Autorização por papéis de usuário (*user role*);
    16.2. Definição de quais transações são críticas (precisam de criptografia) e não-críticas;
17. Manutenção e evolução em granularidade fina.

4 Duas Propostas Existentes

Diante da crescente necessidade de um processo de desenvolvimento voltado para aplicações web, ao longo dos últimos anos, alguns processos específicos e evoluções de processos de software tradicionais foram propostos para web.

A seguir, serão brevemente descritos dois processos. Uma proposta ágil chamada *XWebProcess* e uma proposta mais rigorosa chamada *OPEN-Web Process*.

4.1 XWebProcess

XWebProcess [Sampaio, 2004] [Sampaio et al, 2004a] [Sampaio et al, 2004b] é um processo ágil para desenvolvimento de aplicações web baseado nos princípios do XP (*Extreme Programming*) [Beck, 2000]. O *XWebProcess* é resultado da adaptação do XP para lidar melhor com importantes questões de sistemas web: interfaces com usuário complexas, navegação, requisitos não-funcionais (distribuição, concorrência, balanceamento de carga), testes e suporte de infra-estrutura.

Uma vez que o *XWebProcess* deriva do XP, é interessante, em primeiro lugar, apresentar uma breve descrição do XP. O processo ágil XP confia em cinco pilares de valor:

- Comunicação: significa que os membros da equipe devem interagir constantemente para discutir os problemas e propor soluções. Além disso, o cliente, o alguém que o represente, deve ser um membro ativo da equipe para elucidar os requisitos e regras de negócio envolvidos;
- *Feedback*: implica que cada membro da equipe (programador, gerente, cliente, etc.) deve fornecer *feedback* constante sobre os problemas encontrados para resolvê-los o quanto antes;
- Simplicidade: consiste em escolher a solução mais simples que funcione, ou seja, evitar complexidade e trabalho desnecessário. Fazer estritamente o necessário e com o design mais simples. Possíveis problemas que surjam em função desta decisão são minimizados com práticas como *refactoring* e testes automatizados;
- Coragem: é preciso ter coragem para substituir o que está ruim ou errado por algo que esteja melhor ou correto. Ou seja, não tenha medo de fazer *refactoring* em grandes partes do código quando necessário, bem como descartar requisitos que se tornem obsoletos.
- Respeito: os membros da equipe têm que respeitar uns aos outros. No XP, programadores nunca devem confirmar modificações que façam o programa parar de compilar, que façam os testes falharem, ou caso contrário atrasam o trabalho de seus companheiros. Devem sempre primar pela qualidade. Sob

outra ótica, nenhum membro da equipe deve se sentir rejeitado ou ignorado, de forma a manter a equipe sempre feliz e motivada.

Para subsidiar os quatro pilares acima, XP propõe algumas práticas que são, resumidamente, descritas a seguir:

- *Planning game* (Jogo de Planejamento): um *release* deve ser o menor possível (dois a três meses). Cada *release* é dividido em iterações semanais (duas a três semanas), onde historietas (pequenos casos de uso) são implementados. No início da semana, desenvolvedores e cliente reúnem-se para analisar as funcionalidades. O esforço estimado para implementar cada historieta é definido pelos programadores e o cliente define a prioridade da historieta.
- *Small Releases* (Pequenas Versões): um *release* deve ser pequeno de forma a facilitar o desenvolvimento, bem como o processo de aceitação e satisfação por parte do cliente.
- *Metaphor* (Metáfora): metáfora do sistema é uma história que todos (clientes, programadores, gerentes, etc.) podem contar sobre como o sistema funciona. Enfim, uma história que facilite o entendimento comum sobre os conceitos (abstrações) envolvidos.
- *Simple Design* (Projeto Simples): projetar, de forma simples, estritamente o necessário sob demanda.
- *Whole Team* (Equipe Coesa): o cliente deve ser um membro efetivo da equipe, disponível para esclarecer eventuais questionamentos.
- *Customer Tests* (Testes de Aceitação): testes construídos pelo cliente, em conjunto com analistas e testadores, para validar um ou mais requisitos do sistema. Também chamados de testes funcionais.
- *Sustainable Pace* (Ritmo Sustentável): a equipe deve trabalhar em um ambiente adequado e agradável, sempre motivada e sob um ritmo de trabalho saudável (40 horas/semana, 8 horas/dia) sem horas extras. Horas extras somente se forem impreterivelmente necessárias.
- *Stand*-up *Meeting* (Reuniões em Pé): reuniões em pé (por exemplo, no início do dia) para não perder o foco, somente abordando o que foi realizado e próximas tarefas a realizar. Estas reuniões devem ser de curta duração.
- *Collective Ownership* (Posse Coletiva): o código fonte é um bem comum, ou seja, qualquer membro da equipe pode alterar qualquer parte do código sem precisar pedir permissão. “Todos” devem ter conhecimento de “todas” as partes do código.
- *Pair Programming* (Programação em Par): todo código é produzido por um par de programadores. Um programador codifica e outro analisa, sugerindo melhoramentos e prevenindo erros. Os pares são freqüentemente modificados, fazendo com que os programadores participem do desenvolvimento de diferentes historietas, obtendo, desta forma, um conhecimento geral de todo o sistema;
- *Coding Standards* (Padrões de codificação): devem ser definidos padrões (regras) de estruturação do código. Estas regras devem ser seguidas, garantindo assim uma homogeneidade em todo código, facilitando o entendimento.

- *Test Driven Development* (Desenvolvimento Orientado a Testes): primeiro crie os testes unitários e, somente depois, crie código que passe nos testes. Tal prática ajuda a manter a qualidade do projeto.
- *Refactoring* (Refatoração): o código deve ser, sempre que necessário, melhorado, sem alterar a funcionalidade. Juntamente com testes automatizados (*automated testing*) ajuda a manter a qualidade do código, dado que toda vez que ocorre uma alteração todos os testes, na íntegra, têm que ser passados com sucesso. Toda funcionalidade pode ser alterada por qualquer membro da equipe.
- *Continuous Integration* (Integração Contínua): a equipe de desenvolvimento deve trabalhar sempre na última versão do projeto. Sempre que uma nova funcionalidade for produzida e passar nos testes, esta deve ser integrada a versão atual do sistema, no repositório de código. Isto evita atrasos mais tarde devido a problemas de integração.

O processo *XWebProcess* foi criado adaptando elementos do XP e adicionado novos elementos de forma a customizar o XP para o domínio web. Primeiramente o XP foi modelado usando o meta-modelo SPEM [SPEM] para melhor entendimento. Em seguida, com base nas características das aplicações web, o modelo SPEM do XP foi enriquecido gerando o modelo SPEM do *XWebProcess*. Sendo assim o *XWebProcess* pode ser visto como uma extensão do XP, como mostra a figura 1.

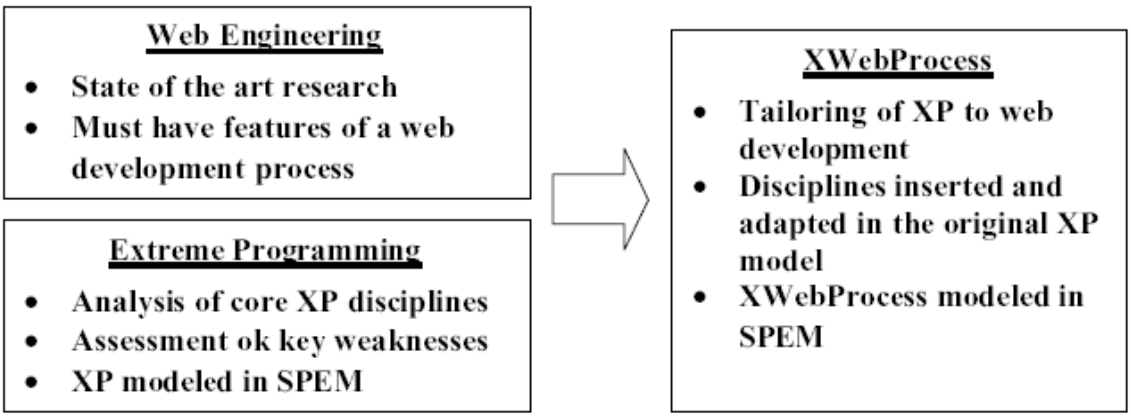

Figura 1. Passos de criação do *XWebProcess* [Sampaio et al, 2004a]

O *XWebProcess* é descrito, em SPEM, usando duas visões: visão dinâmica e estrutural. A visão dinâmica mostra como as disciplinas[1] estão relacionadas entre si durante a execução do processo, ao longo do tempo. Já a visão estrutural descreve a estrutura interna de cada disciplina, destacando atividades, artefatos produzidos e atores envolvidos, bem como os relacionamentos entre eles.

A figura 2 ilustra a visão dinâmica do *XWebProcess*. Nesta figura, as disciplinas em destaque são disciplinas inseridas ou modificadas no XP.

1 Uma disciplina em SPEM representa um conjunto de elementos de processo relacionados (atividades, artefatos, atores) agrupados por um tema comum. Exemplos de disciplinas são: análise, design, testes, etc.

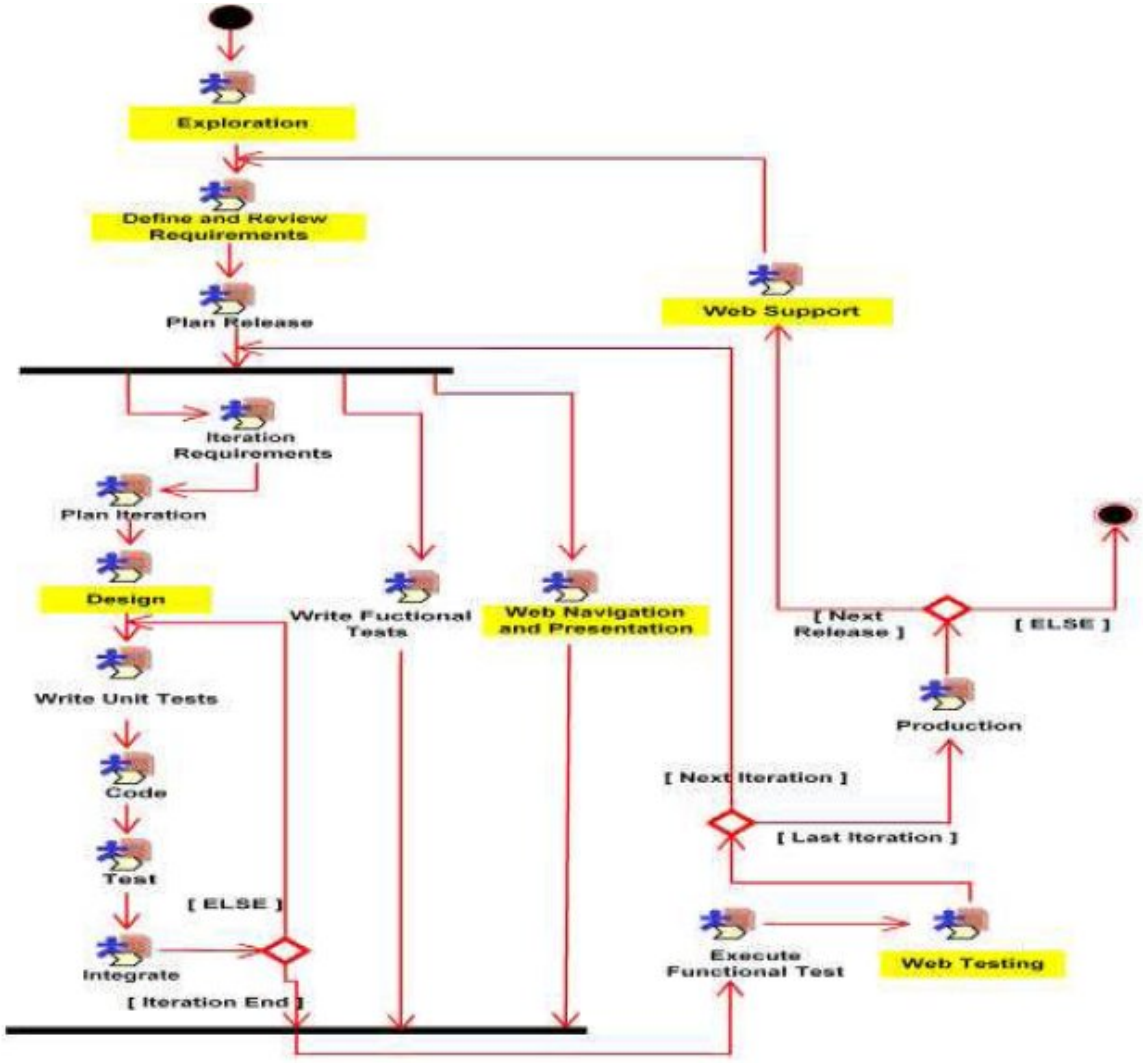

Figura 2. Visão dinâmica do *XWebProcess* [Sampaio et al, 2004a]

O processo começa com uma disciplina exploratória que foi modificada para incluir sessões de protótipo. O objetivo é avaliar, junto ao cliente e analistas de negócio, tecnologias, arquiteturas e protótipos para verificar a viabilidade do projeto e definir os requisitos iniciais.

Em seguida, temos a disciplina de definição e revisão dos requisitos. Nesta, clientes e programadores definem as historietas para o próximo *release*. Programadores estimam o esforço necessário para cada historieta e os clientes definem as prioridades das historietas. Esta disciplina foi modificada para incluir uma atividade de design de arquitetura, visando definir uma estrutura flexível que permita, mais facilmente, integração com outros sistemas e adição de novas tecnologias.

Com as historietas em mãos, clientes e programadores planejam o próximo *release*, selecionando as historietas a serem implementadas.

Dentro de cada *release*, várias interações ocorrem. Em cada iteração, historietas são implementadas e testadas. Para cada iteração ocorrem as seguintes disciplinas: planejamento de iteração, design, escrita de teste de unidade, codificação, teste e integração. A disciplina de design foi modificada para incluir a atividade de design da camada de dados. Durante uma iteração, pode haver mudanças nos requisitos e estimativas anteriores precisam ser reavaliadas. Escrita de testes funcionais, bem como design de na-

vegação e apresentação são feitos em paralelo com as atividades anteriores. A disciplina de design de navegação e apresentação foi inserida devido à inquestionável importância que estes dois aspectos têm em uma aplicação web.

Quando a iteração termina, é necessário verificar se o que foi implementado esta em conformidade com o que se esperava. Portanto, são executados testes funcionais para tal propósito. Além disso, foi inserida a disciplina de testes web para verificar se os requisitos não funcionais foram atendidos.

Se for a última iteração do *release*, a versão corrente do sistema é posta em produção. Depois do primeiro *release* do sistema, entre em cena a disciplina de suporte web que foi inserida com o propósito de lidar com a organização de componentes de hardware e software que fazem parte do *web site*.

O processo termina quando todas as historietas tiverem sido implementadas e postas em produção.

Tendo descrito a dinâmica do processo, a seguir será apresentada a estrutura interna de cada disciplina modificada ou inserida no XP, ou seja, as disciplinas em destaque da figura 2. Para cada disciplina será apresentada um modelo, similar a um diagrama de classes, usando estereótipos do SPEM. Novamente, em cada figura, serão destacados os elementos que foram modificados ou inseridos no XP.

As disciplinas de exploração e definição de requisitos são expostas pelas figuras 3 e 4, respectivamente. Na disciplina de exploração foram incluídas sessões de protótipo. O *web designer* é responsável pela atividade de prototipagem cuja saída é um protótipo, criado com ajuda de técnicas de prototipagem. Já a disciplina de requisitos foi modificada para incluir a definição da arquitetura geral da aplicação. Esta arquitetura é muito importante, pois é usada em todas as atividades subseqüentes. Quem define o modelo de arquitetura é um desenvolvedor experiente que entenda de reuso, manutenção, flexibilidade e performance.

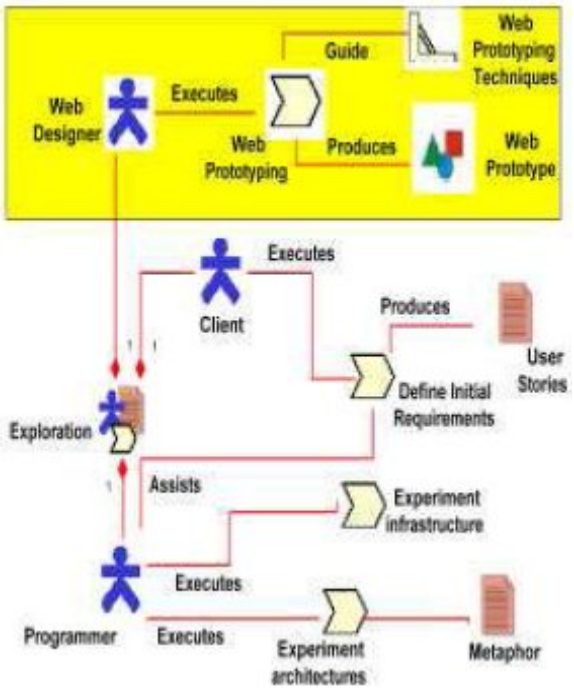

Figura 3. Exploração no *XWebProcess* [Sampaio et al, 2004a]

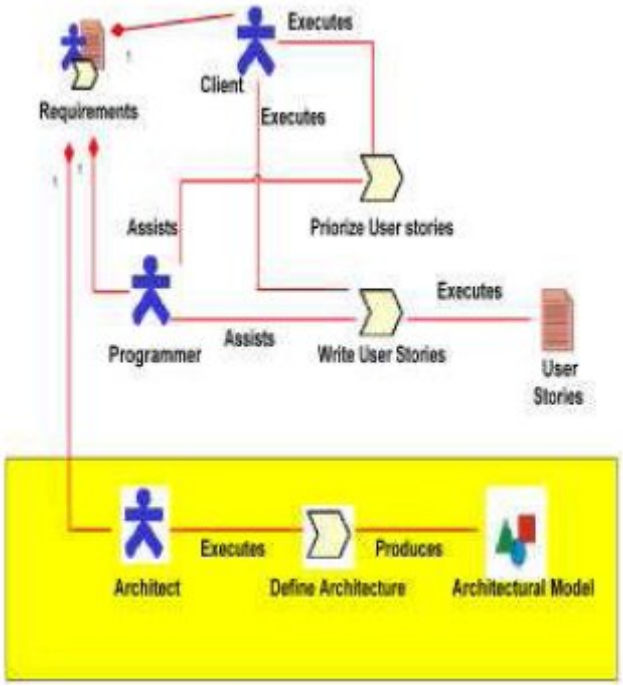

Figura 4. Requisitos no *XWebProcess* [Sampaio et al, 2004a]

A disciplina de análise e design, apresentada na figura 5, foi modificada para incluir a atividade de design de camada de dados, executada pelo DBA. Esta atividade pode ser tão simples quanto definir o esquema de um único banco de dados relacional ou tão complexa quanto fazer a mediação entre várias fontes de dados heterogêneas.

A figura 6 apresenta uma das mais importantes disciplinas inseridas: navegação e apresentação. O *web designer* elabora o design de navegação, assistido pelo programador e arquiteto. Para projetar a navegação, métodos como OOHDM [Schwabe e Rossi, 1995] [Schwabe e Rossi, 1998] são muito bem vindos, pois possuem primitivas com este propósito específico. O *web designer* também é responsável por projetar a interface abstrata e, com base nesta, a interface física.

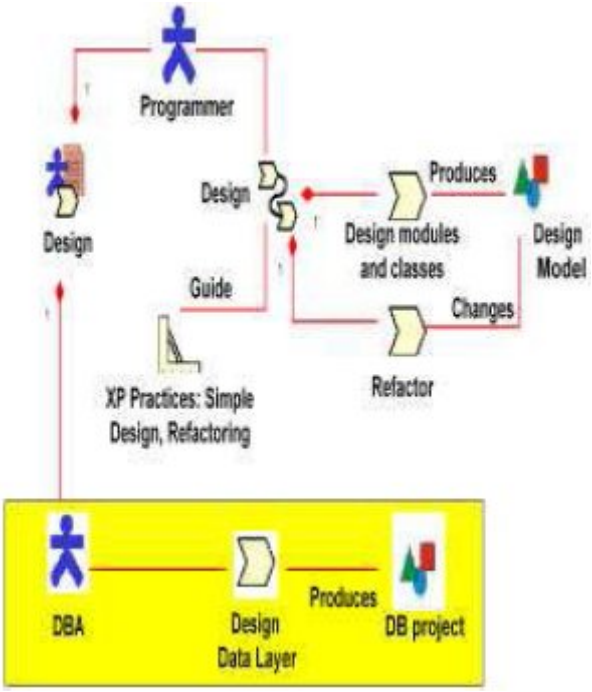

Figura 5. Análise e Design no *XWebProcess* [Sampaio et al, 2004a]

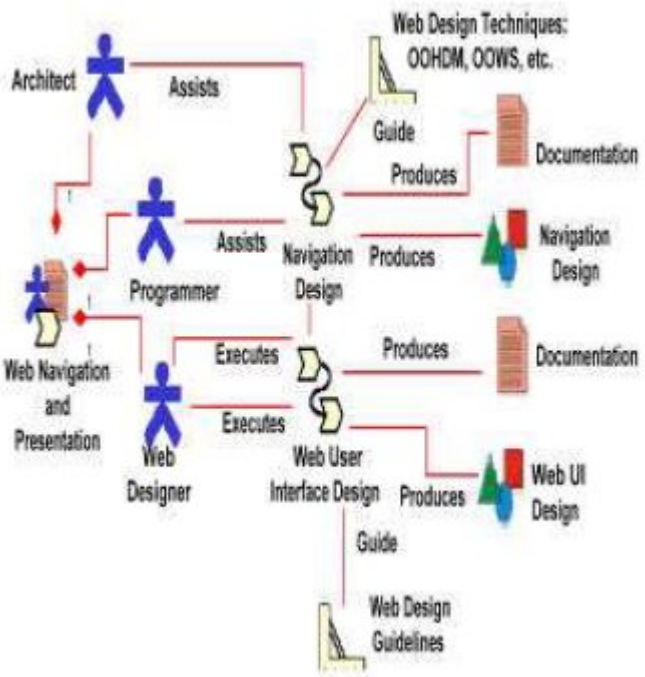

Figura 6. Navegação e Apresentação no *XWebProcess* [Sampaio et al, 2004a]

Para finalizar, as disciplinas de testes web e suporte web são mostradas na figuras 7 e 8, respectivamente. Na disciplina de teste, um programador executa os testes, podendo ser assistido pelo cliente, que os valida, e pelo analista de suporte que configura todo o ambiente para que os testes possam ser realizados. Já na disciplina de suporte, o administrador do *web site* realiza duas atividades: gerenciamento de conteúdo e gerenciamento de infra-estrutura (scripts, páginas, componentes, vídeos, áudio, etc.).

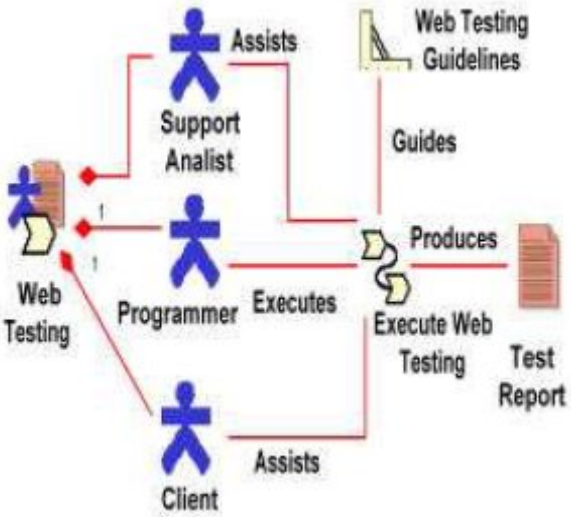

Figura 7. Testes Web no *XWebProcess* [Sampaio et al, 2004a]

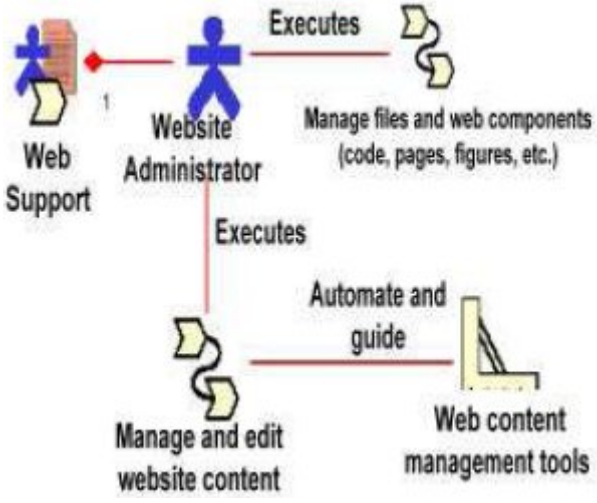

Figura 8. Suporte Web no *XWebProcess* [Sampaio et al, 2004a]

4.2 OPEN-Web Process

OPEN-Web Process [Haire et al, 2001] [Lowe e Henderson-Sellers, 2001] é uma extensão do processo OPEN [OPEN] [Graham et al, 1997] [Henderson-Sellers et al, 1998] para desenvolvimento de aplicações web.

OPEN (*Object-Oriented Process, Environment, and Notation*) é um framework de processo, conhecido por OPF (*OPEN Process Framework*) que abrange o ciclo de vida completo de desenvolvimento, incluindo aspectos de negócio e de software. Trata-se de um meta-processo, a partir do qual pode ser gerado um processo específico (instância) para uma dada organização. OPEN foi desenvolvido e é mantido por um consórcio, sem fins lucrativos, composto por pesquisadores, acadêmicos, desenvolvedores e empresas.

Como pode ser visto na figuras 9 e 10, os maiores componentes deste meta-modelo são:

- *Work Products* (artefatos): os componentes que são desenvolvidos no projeto (documento, diagrama, modelo, módulo, classe, etc.);
- *Languages* (linguagens): os componentes usados para documentar os *work products* (linguagem natural, UML, Java, etc.);
- *Producers* (produtores): os componentes que desenvolvem os *work products* (programador, analista, designer, gerente, cliente, etc.);
- *Work Units* (unidades de trabalho): os componentes que modelam as operações executadas pelos *producers* ao desenvolver *work products*. Há três tipos de *work unit*:
    - *Activity* (atividade): objetivo maior. Define “o quê”, não “como”. É composto por *tasks*;
    - *Task* (tarefa): objetivo menor, ou seja, metas para atingir o objetivo maior. Ainda define “o quê”, não “como”. Resultam na criação, modificação ou avaliação de um ou mais *work products*;
    - *Technique* (técnica): define como perfazer uma dada *task* (casos de uso, cartões CRC, *design patterns*, entrevista, etc.).
- *Stages* (estágios): os intervalos de tempos que provêem uma macro-organização para as *work units*. Podem ter duração (*Cycle, Phase, Workflow, Project, Build, Release, Deployment*) ou não (*milestone*). *Milestone* é um ponto no tempo que indica que uma meta foi atingida. A figura 11 mostra um exemplo de *stages*.

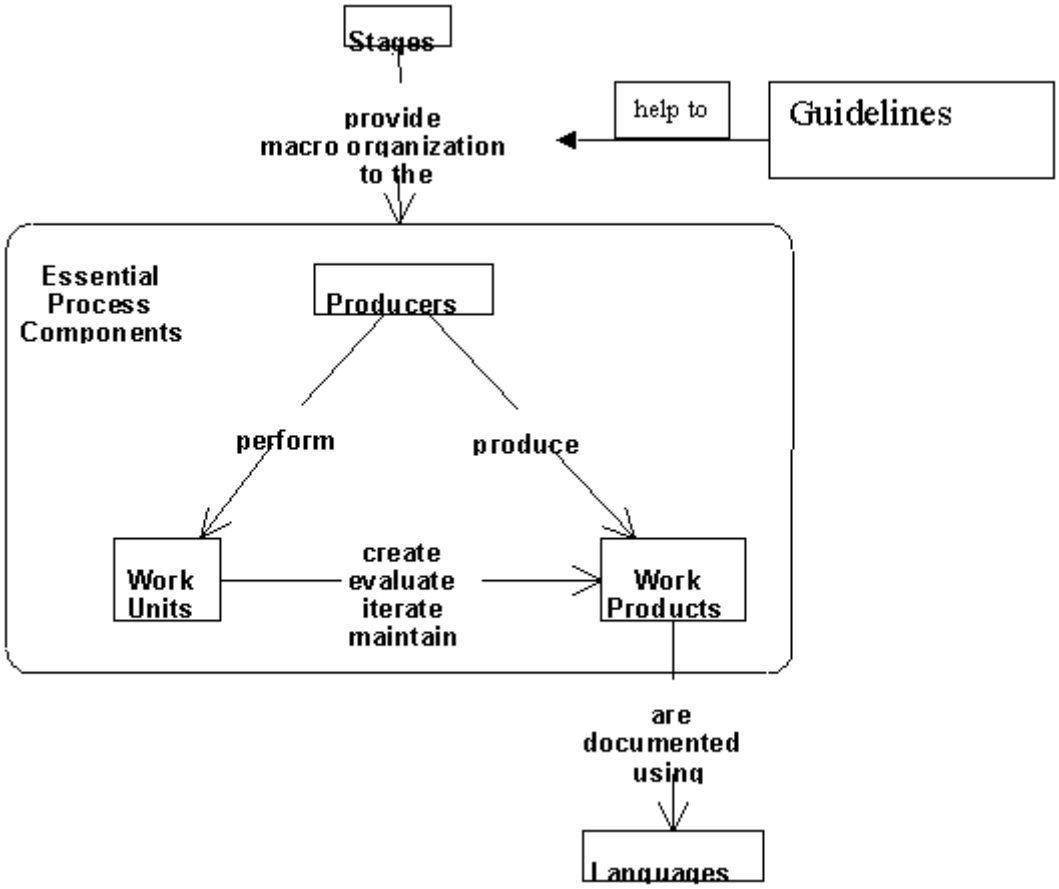

Figura 9. Componentes do meta-processo OPEN [OPEN]

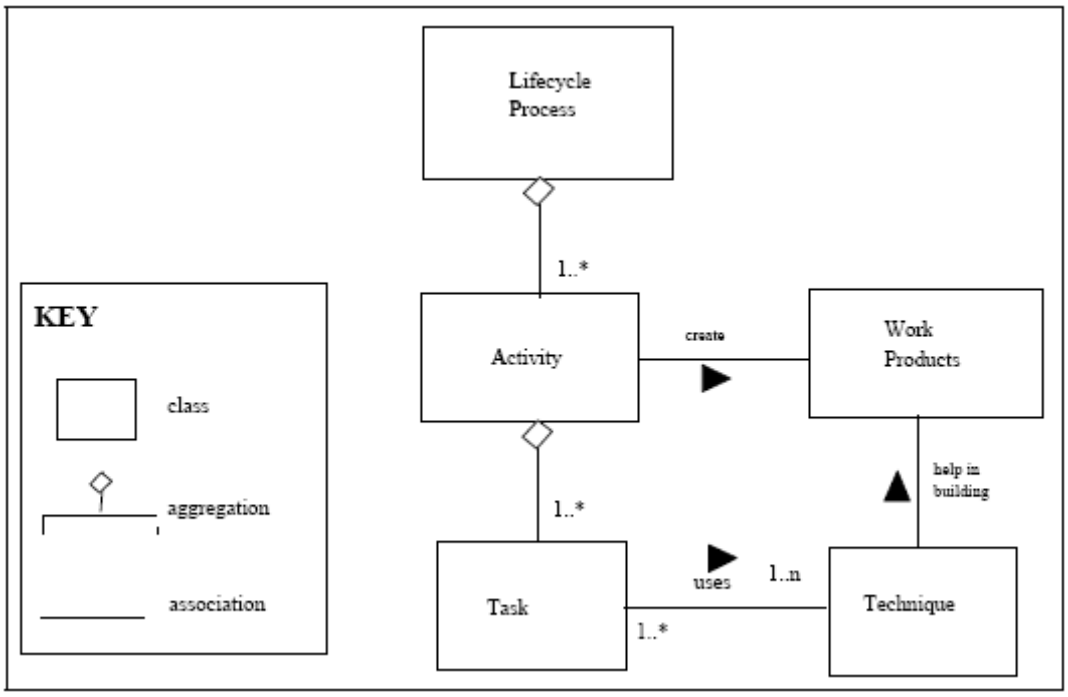

Figura 10. Work Units do meta-processo OPEN [Lowe e Henderson-Sellers, 2001]

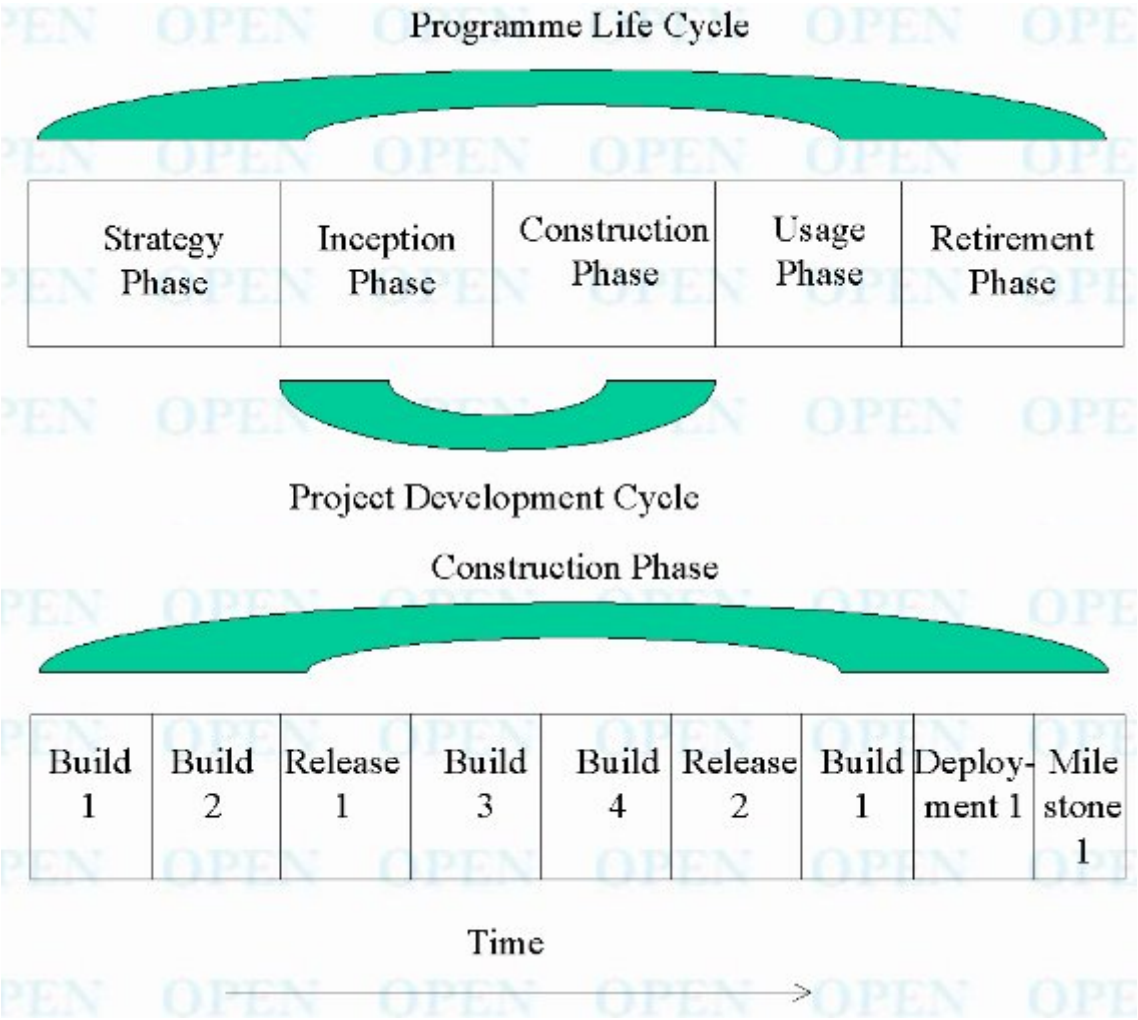

Figura 11. Exemplo de *Stages* do OPEN [OPEN]

Cada processo, instância do meta-processo, é criado escolhendo atividades, tarefas e técnicas específicas, com configurações específicas, como descrito na figura 12. Portanto, OPEN provê um alto grau de flexibilidade para as organizações.

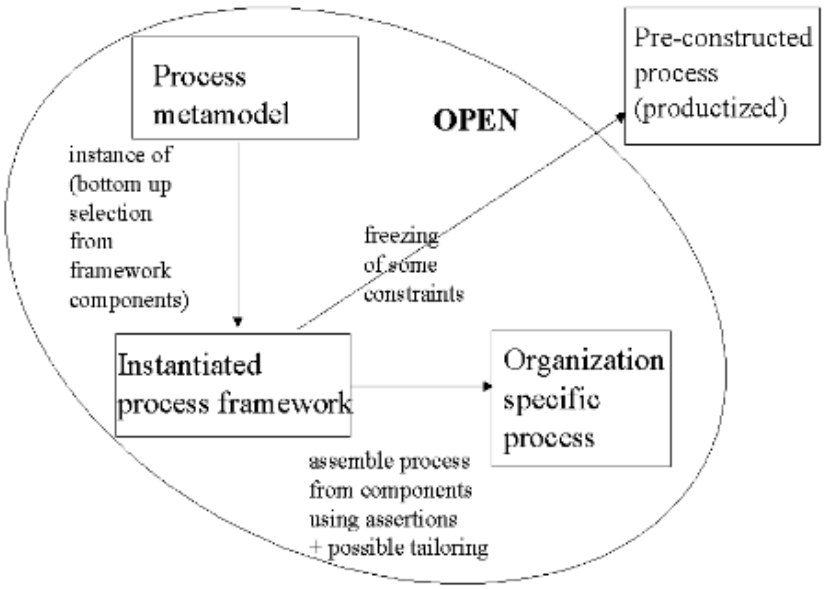

Figura 12. Instanciação do framework OPEN [OPEN]

A natureza baseada em componentes do OPEN permite criar extensões apropriadas para domínios específicos. Desta forma, surgiu o *OPEN-Web Process* com uma extensão voltada para o domínio de aplicações web. Esta extensão foi criada analisando as diferenças entre sistemas web e convencionais e validada usando estudos de casos de projetos comerciais para web.

Apesar das diferenças, certas atividades, tarefas e técnicas do framework OPEN se mantêm relevantes para o desenvolvimento web, sem precisar de alteração. Por exemplo, as tarefas ligadas às atividades de "Iniciação de Projeto", "Planejamento de Implementação" e "Planejamento de Projeto" permaneceram relativamente inalteradas. Estas atividades são as mesmas para qualquer tipo de projeto, ou seja, é necessário obter aprovação, fazer estudos de viabilidade, etc. Já atividades como "Engenharia de Requisitos" e "Implementação" são mais volúveis de acordo com o tipo de projeto.

A fim de criar o *OPEN-Web Process*, foram propostas as seguintes adições e modificações para o framework OPEN:

- Atividade:
  - Gerenciamento de web site.
    - Tratar todas as questões relacionadas ao desenvolvimento, manutenção e gerenciamento de um web site corporativo, que inclua ou não acesso a sistemas de processamento de transação no *back-end*.
- Tarefas:
  - Construir *White Sites* (protótipos);
  - Criar conteúdo
    - Selecionar e preparar todo conteúdo (imagens, vídeos, layout de texto, reuso de *templates*, etc.).
  - Criar mapa de navegação pelo conteúdo;
  - Definir critério de aceitação para o *web site*;
  - Definir estratégias de teste para o *web site*;
  - Projetar e implementar estratégia de gerenciamento de conteúdo;
  - Projetar e implementar estratégia de personalização
    - Customizar conteúdo, apresentação e navegação por perfis de usuário, ou até mesmo fornecer meios para que o próprio usuário possa customizar a aplicação em tempo de execução.
  - Projetar arquitetura do *web site*;
  - Projetar padrões de *web site*
    - Fazer uso de padrões para garantir consistência, quer seja sob a perspectiva de usabilidade, quer seja sob a perspectiva de desenvolvimento. O *World Wide Web Consortium* [W3C] tem feito consideráveis contribuições neste sentido.
  - Desenvolver uma identidade para o *web site*
    - Desenvolver um estilo próprio do site que esteja estrategicamente ligado com a missão do site e que seja marcante para o público.

- Desenvolver um padrão de dados;
    - Facilita o inter-câmbio de dados entre componentes de um mesmo sistema e, em especial, em interações B2B. Novamente o *World Wide Web Consortium* tem trabalhado muito na concepção de padrões XML para a indústria que podem ser muito úteis.
- Integrar conteúdo com interface com usuário;
- Criar protótipos de interface com usuário;
- Realizar gerenciamento de conteúdo;
- Realizar análise de mercado;
    - Procurar conhecer o público alvo para definir a viabilidade do projeto e também para oferecer um serviço mais apropriado.
- Testar o *web site*
    - Em especial requisitos não funcionais de qualidade como performance e concorrência.

- Sub-tarefas:
    - Escolher um padrão arquitetural para o web site
        - Sub-tarefa de “Definir uma arquitetura”.
    - Criar o modelo de papéis ou atores (*UCD*[2] *role model*)
        - Sub-tarefa de “Projetar Interface do Usuário”.
    - Criar o modelo de tarefas (*UCD task model*)
        - Sub-tarefa de “Projetar Interface do Usuário”.
    - Criar o modelo de conteúdo (*UCD content model*)
        - Sub-tarefa de “Projetar Interface do Usuário”.
- Tarefas relacionadas ao desenvolvimento baseado em componentes:
    - Escolher um framework apropriado;
    - Avaliar o potencial de frameworks;
    - Integrar componentes;
    - Enumerar uma lista de frameworks candidatos.

2 UCD – *Usage Centered Design*. Diferentemente de design baseado no usuário, trata-se de design baseado no trabalho que o usuário irá realizar e no que o software precisa fornecer, via interface com o usuário, para ajudá-lo a realizar.

5 Avaliação das Propostas Apresentadas

Esta secção apresenta uma avaliação das duas propostas apresentadas de acordo com os requisitos que um processo de software para desenvolvimento de aplicações deve atender. Cada proposta é avaliada com relação a cada requisito, indicando o quão completo a proposta contempla tal requisito, sob o seguinte esquema: não (não contempla), parcial (contempla parcialmente), total (contempla totalmente).

Avaliação do *XWebProcess*:

1. Autoria (ou engenharia) de conteúdo: não.
   - Não possui uma atividade explícita para preparação do conteúdo. Presume-se que seja realizado algo neste sentido ao fazer o design de classes, ou seja, modelo conceitual de domínio.
2. Autoria (ou engenharia) de navegação sobre o conteúdo: total.
   - Atividade de "Design de Navegação" dentro da disciplina "Navegação e Apresentação".
3. Autoria (ou engenharia) de apresentação (interface): total.
   - Atividade de "Design de Interface com o Usuário" dentro da disciplina "Navegação e Apresentação"
4. Ubiqüidade: não.
5. Definição de arquitetura multicamadas: parcial.
   - Atividade "Definir Arquitetura" dentro da disciplina "Definição e Revisão de Requisitos". Todavia, deveria possuir sub-atividades mais específicas relacionadas ao uso de padrões arquiteturais, frameworks, sistema legados, distribuição entre camadas e separação entre o que roda no navegador cliente e o que roda no servidor web.
6. Ciclos de vida de desenvolvimento curtos: total.
   - Esta característica foi herdada do XP.
7. Equipes multidisciplinares: total.
   - Inclui o cliente e os outros papéis relacionados ao desenvolvimento, bem como menciona a figura de analista de negócio e *expert* de domínio.
8. Desenvolvimento concorrente de pequenas equipes em tarefas interdependentes: não.
   - Não menciona atividades em paralelo e coordenação de várias pequenas equipes.
9. Pesquisa de mercado: não.
   - Não alude.
10. Incerteza do cliente e volatilidade dos requisitos: parcial.
    - Na disciplina de "Exploração" foram incluídas sessões de protótipo para lidar com a incerteza. Também é mencionada, na disciplina "Definição e

Revisão de Requisitos", a necessidade de definição de uma arquitetura flexível que admita mudanças com mais facilidade. No entanto, não fica claro como tratar mudanças nos requisitos à medida que o sistema evolui.

11. Reengenharia de modelos de negócio a partir de modelos da aplicação: não.
    - Não oferece suporte para análise de impactos de modelos de software no negócio.
12. Análise de negócio e avaliação junto ao usuário final: parcial.
    - Análise de negócio pode ser vista na disciplina de "Exploração" e de "Definição e Revisão de Requisitos", no entanto não faz referência ao envolvimento do usuário final ao longo do ciclo de desenvolvimento antes de colocar a aplicação em produção.
13. Diferentes perfis de usuário (personalização): não.
    - Não faz referência.
14. Requisitos explícitos (funcionais e não-funcionais): total.
    - Na disciplina "Definição e Revisão de Requisitos".
15. Testes rigorosos com relação aos requisitos: total.
    - Na disciplina "Testes Web".
16. Autoria (engenharia) de Segurança: não.
    - Não alude.
17. Manutenção e evolução em granularidade fina: parcial.
    - Na disciplina "Suporte para Web", por meio da atividade gerenciamento de conteúdo e de componentes. Entretanto, não deixa explícito como fazer, quais os passos a serem seguidos.

Avaliação do *OPEN-Web Process*:

1. Autoria (ou engenharia) de conteúdo: total.
    - Tarefa "Criar conteúdo".
2. Autoria (ou engenharia) de navegação sobre o conteúdo: total.
    - Tarefa "Criar mapa de navegação pelo conteúdo".
3. Autoria (ou engenharia) de apresentação (interface): total.
    - Tarefa "Projetar Interface do Usuário" e suas sub-tarefas. Além destas, as tarefas "Integrar conteúdo com interface com usuário" e "Criar protótipos de interface com usuário".
4. Ubiqüidade: não.
5. Definição de arquitetura multicamadas: parcial.
    - Atividade "Projetar arquitetura do *web site*" e sua sub-tarefa "Escolher um padrão arquitetural", bem como as tarefas relacionadas ao uso de componentes e frameworks. Não obstante, deveria possuir sub-atividades mais

específicas relacionadas ao uso de sistema legados, distribuição entre camadas e separação entre o que roda no navegador cliente e o que roda no servidor web.

6. Ciclos de vida de desenvolvimento curtos: não.
    - O meta-processo OPEN requer um engenheiro de software muito bom que conduza sua instanciação para um processo de software particular. Desta forma, existe uma dependência muito grande na capacidade do engenheiro de software em definir um processo que atenda aos prazos curtos exigidos por uma aplicação web.
7. Equipes multidisciplinares: parcial.
    - Inclui o cliente e os outros papéis relacionados ao desenvolvimento, mas não menciona a figura de analista de negócio e *expert* de domínio.
8. Desenvolvimento concorrente de pequenas equipes em tarefas inter-dependentes: não.
    - Não menciona atividades em paralelo e coordenação de várias pequenas equipes.
9. Pesquisa de mercado: total.
    - Tarefa "Realizar análise de mercado".
10. Incerteza do cliente e volatilidade dos requisitos: parcial.
    - Nas tarefas "Construir *White Sites*" e "Criar protótipos de interface com usuário" são criados protótipos para lidar com a incerteza. Também é mencionado, na tarefa "Projetar uma arquitetura" e suas sub-tarefas, a necessidade de definição de uma arquitetura flexível que admita mudanças com mais facilidade. Também estão previstas duas técnicas: "*Development spikes*" e "*Field trips*". A primeira é utilizada para ajudar aos clientes a compreenderem a tecnologia e o domínio do problema através de pequenos protótipos. A segunda consiste em analisar o ambiente do negócio e o lugar final onde será instalada a aplicação. No entanto, não fica claro como tratar mudanças nos requisitos à medida que o sistema evolui.
11. Reengenharia de modelos de negócio a partir de modelos da aplicação: não.
    - Não oferece suporte para análise de impactos de modelos de software no negócio. No entanto o framework OPEN prevê uma fase (*phase*) chamada Reengenharia do Negócio.
12. Análise de negócio e avaliação junto ao usuário final: parcial.
    - O framework OPEN inclui modelagem de negócio, mas não deixa claro como isto se relaciona com análise de negócio. O *OPEN-Web Process* não menciona uma tarefa de avaliação ou o envolvimento do usuário final ao longo do desenvolvimento. No entanto, possui várias tarefas baseadas em UCD.
13. Diferentes perfis de usuário (personalização): total.
    - Tarefa "Projetar e implementar estratégia de personalização".
14. Requisitos explícitos (funcionais e não-funcionais): total.

- Na atividade “Engenharia de Requisitos”, bem como na tarefa “Definir critério de aceitação para o *web site*”.

15. Testes rigorosos com relação aos requisitos: total.
    - Nas tarefas “Definir critério de aceitação para o *web site*”, “Definir estratégias de teste para o *web site*” e “Testar o *web site*”.
16. Autoria (engenharia) de Segurança: não.
    - Não faz menção.
17. Manutenção e evolução em granularidade fina: parcial.
    - Na atividade “Gerenciamento do *web site*”, por meio da atividade gerenciamento de conteúdo e de componentes. Entretanto, não deixa explícito como fazer, quais os passos a serem seguidos.

Os resultados da avaliação estão sumarizados na tabela 1. Percebe-se que nenhum dos dois processos atende plenamente a todos os requisitos. Isto é uma evidência que ainda há a necessidade de criação ou aprimoramento de processos que apóiem a elaboração de aplicações web, sendo, portanto, uma ampla avenida para projetos de pesquisa.

Requisito \ Processo	XWebPrcess	OPEN-Web Process
01 Autoria de conteúdo	não	total
02 Autoria de navegação	total	total
03 Autoria de interface	total	total
04 Ubiqüidade	não	não
05 Arquitetura multicamadas	parcial	parcial
06 Desenvolvimentos curtos	total	não
07 Equipes multidisciplinares	total	parcial
08 Desenvolvimento concorrente	não	não
09 Pesquisa de mercado	não	total
10 Incerteza dos requisitos	parcial	parcial
11 Reengenharia do negócio	não	não
12 Análise do negócio	parcial	parcial
13 Personalização	não	total
14 Requisitos explícitos	total	total
15 Testes	total	total
16 Autoria de Segurança	não	não
17 Manutenção	parcial	parcial

Tabela 1: Sumário da avaliação dos processos de acordo a lista de requisitos

6 Considerações Finais

Ao longo da história da engenharia de software vários processos de software foram propostos e certamente muito outros estão por vir. Todos os processos definem de alguma forma um conjunto organizado de práticas a ser seguido de maneira a aumentar o controle durante todo o ciclo de vida do software e, por conseguinte, elevar a qualidade do produto. No entanto, determinados domínio de aplicação, como aplicações web, possui suas especificidades, demandando um processo de desenvolvimento mais especializado.

Este trabalho apresentou as diferenças relevantes entre sistema web e sistemas convencionais a fim de chamar a atenção para a necessidade de criação de um processo de software voltado para aplicações web. Com base nestas diferenças, foi identificada uma série de requisitos que precisam ser levados em consideração.

Com base nos requisitos levantados, foram analisados dois processos existentes direcionados para web. Ambos os processos têm sua importante dose de contribuição, mas nenhum deles prevê integralmente os requisitos identificados. Esta constatação reforça o fato que, embora as aplicações web sejam reconhecidamente um domínio de suma importância em nossos dias, ainda existem muitas práticas ad-hoc de desenvolvimento para suprir a falta de um processo sistemático que preveja certas peculiaridades da engenharia para web.

Como foi visto, uma aplicação web é marcada por incerteza e volatilidade. O cliente não sabe ao certo aonde quer chegar. Ele vai adquirindo este discernimento à medida que a aplicação evolui. Além disso, a tecnologia está em constante evolução fazendo com que a solução projetada precise ser flexível o bastante para absorver estas mudanças. Outro fato importante é o treinamento da equipe que precisa estar sempre reciclando seu conhecimento.

Diante de todas estas diferenças e desafios, envolvendo várias áreas, desde o nível de negócio até o nível técnico, fica claro que aplicações web merecem uma atenção especial. Definir um processo de software para web certamente não é uma tarefa fácil. Na verdade, trata-se de uma composição de sub-processos, cada processo cobrindo questões particulares de cada área envolvida.

A motivação deste trabalho foi justamente chamar a atenção para esta necessidade e prover um conjunto de requisitos que possa servir como passo inicial para uma iniciativa de definição de processo de software para sistemas web.

Referências

[Agile Manifesto] Agile Manifesto Web Site. http://www.agilemanifesto.org

[Beck, 2000] Beck, Kent. Extreme Programming Explained, Addison Wesley, 2000.

[Bohem e Turner, 2004] Boehm, B.W.; Turner, R. Balancing Agility and Discipline: A guide for the Perplexed, Reading, Massachusetts: Addison-Wesley, 2004.

[Chrissis et al, 2003] Chrissis, M.B.; Konrad, M.; Shrum, S. CMMI: Guidelines for Process Integration and Product Improvement, Addison-Wesley, 2003.

[Graham et al, 1997] Graham, I.; Henderson-Sellers, B.; Younessi, H. The OPEN Process Specification, Addison-Wesley, 1997.

[Haire et al, 2001] Haire, B.; Henderson-Sellers, B.; Lowe, D. “Supporting web development in the open process: additional tasks,” in COMPSAC’2001: International Computer Software and Applications Conference, Chicago, Illinois, USA, Submitted, IEEE Computer Society.

[Henderson-Sellers et al, 1998] B. Henderson-Sellers, A. Simons, and H. Younessi, The OPEN Toolbox of Techniques, Addison-Wesley, UK, 1998.

[Humphrey, 1995] Humphrey, W.S. Personal Software Process, Addison Wesley, 1995.

[Humphrey, 2000] Humphrey, W.S. Introduction to the Team Software Process, New York, Addison-Wesley, 2000.

[Kappel et al, 2004] Kappel, Gerti; Michlmayr, Elke; Pröll, Birgit; Reich, Siegfried; Retschitzegger, Werner. Web Engineering - Old wine in new bottles? International Conference on Web Engineering (ICWE), Munich 2004.

[Lowe e Henderson-Sellers, 2001] Lowe, David B.; Henderson-Sellers, Brian. Characteristics of Web Development Processes, presented at SSGRR 2001, L'Aquila, Italy, 2001.

[McDonald e Welland, 2004] A. McDonald and R. Welland. Evaluation of Commercial Web Engineering Processes. in Fourth International Conference on Web Engineering, ICWE 2004.

[OPEN] OPEN web site: http://www.open.org.au

[Rossi, 1996] Rossi, Gustavo; Schwabe, Daniel. “Um método orientado a objetos para o projeto de Aplicações Hipermídia”, Tese de Doutorado, Departamento de Informática PUC-Rio, 1996.

[Sampaio et al, 2004a] Sampaio, Américo; Vasconcelos, Alexandre; Sampaio, Pedro R. Falcone. Design and Empirical Evaluation of na Agile Web Engineering Process. In 18th Simpósio Brasileiro de Engenharia de Software, SBES 2004.

[Sampaio et al, 2004b] Sampaio, Américo; Vasconcelos, Alexandre; Sampaio, Pedro R. Falcone. XWebProcess: Agile Software Development for Web Applications, Accepted Poster Presentation to appear in International Conference on Web Engineering – ICWE04, Munich Germany, 2004.

[Sampaio, 2004] Sampaio, Américo. XWebProcess: Um processo ágil para o desenvolvimento de aplicações Web. MSc Thesis, Universidade Federal de Pernambuco, Brazil, March 2004.

[Schwabe e Rossi, 1995] Schwabe, Daniel; Rossi, Gustavo. The Object-Oriented Hypermedia Design Model. In Communications of the ACM, volume 38(8), pages 45-46, 1995.

[Schwabe e Rossi, 1998] Schwabe, Daniel; Rossi, Gustavo: “An object-oriented approach to web-based application design”. Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet, v. 4#4, pp.207-225, October, 1998.

[SPEM] Object Management Group. Software Process Engineering Metamodel Specification. Version 2.0 http://www.omg.org/technology/documents/formal/spem.htm

[SPICE] Software Process Improvement and Capability dEtermination. http://www.sqi.gu.edu.au/spice/

[W3C] World Wide Web Consortium web site: http://www.w3.org