UNIVERSIDADE FEDERAL DE SÃO CARLOS

DEPARTAMENTO DE COMPUTAÇÃO

ENGENHARIA PARA A WEB

FABRICIO APARECIDO BREVE

SÃO CARLOS, JUNHO DE 2002

ÍNDICE

Introdução

O número de sistemas e aplicativos para a *Web* tem crescido muito nos últimos anos, causando um grande impacto na história da computação. Com a sua importância aumentando, também se faz necessário o uso de uma abordagem disciplinada para a construção destes sistemas. Abordagens utilizadas na Engenharia de Software tradicional passaram a ser adotadas na construção de sistemas para a *Web*.

Os aplicativos para a *Web* são diferentes de outras categorias de software e têm características exclusivas: são dirigidos a conteúdo, estão em constante evolução, têm curto prazo de desenvolvimento, dentre outras.

O grande desafio passa a ser adaptar as técnicas existentes para uso na *Web*, criando assim o que é chamado de "Engenharia para a *Web*" (*Web Engineering*). Este trabalho corresponde a um estudo de tais técnicas e da maneira com que elas vêm sendo adotadas pelos desenvolvedores de sistemas para a *Web*.

1. A Evolução da *Web*

A *Web* foi inicialmente concebida com o intuito de compartilhar informações científicas entre alguns poucos cientistas. O conteúdo era estático e apenas textual, não havia imagens, sons, animações ou conteúdo gerado dinamicamente para cada usuário, a interação era limitada, a navegabilidade era fácil, alto desempenho era desejável, mas não essencial, os *sites* eram desenvolvidos por apenas uma pessoa ou um pequeno grupo.

Mas a *Web* evoluiu e hoje ouvimos música, vemos filmes, compramos e vendemos produtos, conhecemos pessoas e inúmeros outros usos que não poderiam ser imaginados dez anos atrás. Murugesan destaca que a Internet levou apenas quatro anos para estar em 30% dos lares americanos. É um tempo bem curto quando comparado a outros produtos: o telefone levou 40 anos, o rádio levou 35 anos, o videocassete demorou 20 anos, a televisão 26 anos e o próprio computador levou 19 anos. [MUR00] [GIN01b]

Escopo e complexidade foram aumentando, pequenos serviços foram cedendo espaço para grandes aplicativos, e com isso também aumentou a complexidade dos projetos e as dificuldades de desenvolvimento, manutenção e gerenciamento. Tudo feito com pouca disciplina, sem preocupação com técnicas e métodos padronizados ou maneiras de controlar a qualidade.

A atitude predominante era: “Vamos fazer rápido, não há tempo para planejar”, e acabou gerando aplicativos com grande probabilidade de ter problemas como baixo desempenho e falhas. Na *Web* problemas como esses são ainda mais graves que no software tradicional, pois como afirma Nielsen: a distância entre um *site* e seus concorrentes é sempre de apenas poucos “cliques”. [NIE00]

Falta de planejamento, projetos mal feitos e falta de gerenciamento acabam tendo conseqüências sérias. Segundo Ginige e Murugesan, 84% dos sistemas entregues não atendem às necessidades do cliente; 79% dos projetos são entregues com atrasos e 63% têm custo maior que o orçamento previsto. Mais de 50% dos sistemas prontos são de baixa qualidade e faltam funcionalidades necessárias. [GIN01]

Como resultado, desenvolvedores e usuários, começaram a se preocupar com a maneira como sistemas *Web* complexos estão sendo criados, bem como com seus níveis de desempenho, qualidade e integridade. E aí surge a Engenharia para a *Web*.

2. O que é Engenharia para a *Web*

Segundo Pressman, sistemas e aplicativos da *Web* são caracterizados por disponibilizar grande quantidade de conteúdo e funcionalidade para grande população de usuários. A Engenharia para a *Web* é, portanto, o processo utilizado para criar aplicativos *Web* de alta qualidade. A Engenharia para *Web* não é igual à Engenharia de Software tradicional, mas compartilham muitos conceitos e princípios fundamentais, com ênfase nas mesmas técnicas de gerenciamento e atividades. Há pequenas diferenças na maneira como essas atividades são conduzidas, mas a filosofia que dita uma abordagem disciplinada para o desenvolvimento de um sistema de computador é a mesma. [PRE01]

Ao mesmo tempo em que adota muitos princípios da Engenharia de Software, a Engenharia para a *Web* incorpora novas abordagens, metodologias, ferramentas, técnicas e normas para atender os requisitos exclusivos dos sistemas para a *Web*. O desenvolvimento de sistemas para a *Web* é significantemente diferente do desenvolvimento de software tradicional e apresenta vários desafios adicionais. Também é um erro achar que o desenvolvimento de aplicativos para *Web* é apenas a criação de páginas utilizando *HTML*, *FrontPage* ou *Dreamweaver*. E da mesma forma é equivocado achar que desenvolvimento para a *Web* envolve apenas a manipulação de diversas mídias e criação de conteúdo. O desenvolvimento para a *Web* é uma mistura de publicações impressas e desenvolvimento de software, entre marketing e computação, entre comunicações internas e relações externas, e entre arte e tecnologia. [GIN01] [MUR00]

A construção de um sistema para a *Web* necessita do conhecimento de pessoas de diferentes áreas. Como resultado, a Engenharia para a *Web* é multidisciplinar, e dela participam áreas como: análise de sistemas e projetos; engenharia de software; engenharia de hipermídia e hipertexto; engenharia de requisitos; interação humano-computador; desenvolvimento de interface de usuário; engenharia de informação; indexação e recuperação de informações; teste; modelagem e simulação; gerenciamento de projetos; e projeto gráfico e apresentação. [GIN01]

3. Características de Aplicativos para a *Web*

Segundo Pressman, as seguintes características podem ser encontradas na grande maioria dos aplicativos para a *Web*: [PRE01]

- **Rede Intensiva:** Aplicativos para a *Web* utilizam recursos de rede por natureza. Ele está em uma rede e deve atender as necessidades de diversas comunidades de clientes. Ele pode estar disponível na Internet (permitindo comunicação com o mundo todo), em uma *Intranet* (implementando comunicação em uma organização) ou ainda em uma *Extranet* (comunicação inter-redes).
- **Dirigido a Conteúdo.** Em muitos casos, a função primária do aplicativo para a *Web* é usar hipermídia para apresentar textos, gráficos, e vídeo para os usuários.
- **Evolução contínua.** Ao contrário dos aplicativos convencionais que evoluem através de uma série de versões planejadas e lançadas em determinados intervalos de tempo, os aplicativos para a *Web* evoluem continuamente.

Alguns autores costumam comparar a evolução de aplicativos para a *Web* com jardinagem. Desenvolver um *site* consiste em criar uma infra-estrutura (planejar o jardim) e então "plantar" as informações que irão crescer e florescer neste jardim. Com o tempo, o jardim (o *site*) irá evoluir, mudar e crescer. Um bom planejamento inicial permitirá que este crescimento ocorra de uma maneira controlada e consistente.

Pressman cita as principais diferenças entre desenvolver um aplicativo para a *Web* e desenvolver um software tradicional: [PRE01]

- **Imediatismo:** o tempo que um *site* completo precisa ficar pronto pode ser apenas alguns poucos dias ou semanas. Desenvolvedores devem, portanto, utilizar métodos de planejamento, análise, projeto, implementação e teste que estejam adaptados para estes cronogramas comprimidos necessários no desenvolvimento para a *Web*.
- **Segurança:** aplicativos para a *Web* estão disponíveis via rede, é difícil ou até mesmo impossível limitar a população de usuários que irão acessar o aplicativo. Para poder proteger o conteúdo e fornecer métodos seguros de

transmissão de dados é preciso implementar medidas rígidas de segurança no aplicativo e na infra-estrutura do mesmo.

- **Estética:** é inegável que boa parte do apelo dos aplicativos para a *Web* é o seu visual. Quando um aplicativo é projetado para vender produtos ou idéias, estética pode ser tão importante para o sucesso quanto o projeto técnico.

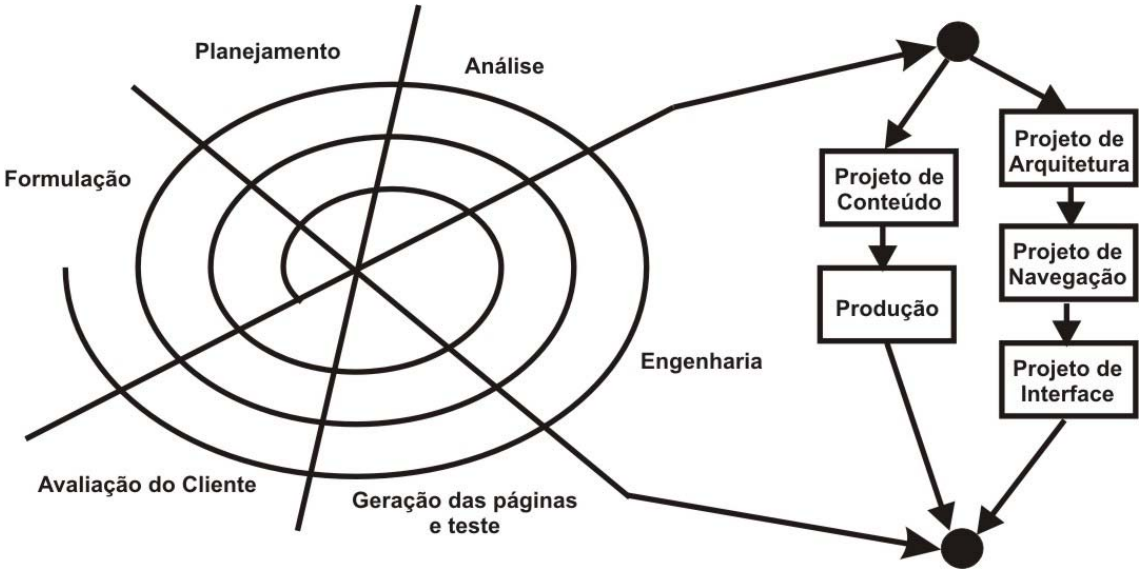
Estas características gerais se aplicam a todos os aplicativos para *Web*, mas com diferentes graus de influência. Podemos categorizar os aplicativos *Web* desta forma:

- *Informacional.* Conteúdo apenas para leitura é fornecido com navegação simples e *links*.
- *Download*. Um usuário faz o *download* de informações dos servidores apropriados.
- *Personalizável.* O usuário personaliza o conteúdo para suas necessidades específicas.
- *Interação*. Comunicação entre uma comunidade de usuários ocorre em salas de bate-papo, fóruns ou mensagens instantâneas.
- *Entrada de Usuário*. Entradas baseadas em formulários são os mecanismos primários para a comunicação necessária.
- *Orientado a transações*. O usuário faz um pedido que é atendido pelo aplicativo.
- *Orientado a serviços*. O aplicativo fornece um serviço para o usuário.
- *Portal*. O aplicativo direciona o usuário para outros conteúdos ou serviços fora do domínio do portal do aplicativo.
- *Acesso a Banco de Dados*. O usuário faz uma consulta em um banco de dados e extrai informações.
- *Data warehousing*. O usuário consulta uma coleção de grandes bancos de dados e extrai informações.

A chave para o sucesso é conviver com as restrições impostas por estas características e mesmo assim construir um bom aplicativo.

4. Modelo de Processo

À medida que os aplicativos para a *Web* evoluem de estáticos para dinâmicos é necessário aplicar um gerenciamento sólido e os princípios de engenharia passam a ter fundamental importância. É necessário, portanto, desenvolver um modelo de desenvolvimento que atenda tais requisitos de forma eficiente. Pressman propõe o seguinte modelo: (PRE01)



O processo começa com a *formulação* – uma atividade que identifica as metas e objetivos do aplicativo e determina um escopo para o primeiro incremento. O *planejamento* estima os custos do projeto, avalia riscos associados com o desenvolvimento, elabora um cronograma bem definido para o primeiro incremento, e um menos definido para os demais. Na *Análise* são estabelecidos os requisitos técnicos e identificados os itens de conteúdo que serão incorporados. Requisitos para projeto gráfico (estética) também são definidos.

A atividade de *Engenharia* incorpora duas atividades paralelas ilustradas no lado direito da figura. *Projeto de Conteúdo* e *Produção* são tarefas desenvolvidas pelos membros "não-técnicos" da equipe. O objetivo destas tarefas é projetar, produzir e/ou obter todos os textos, gráficos, conteúdo de áudio e vídeo que serão integrados ao aplicativo. Ao mesmo tempo, uns conjuntos de tarefas técnicas de projeto são conduzidos.

Geração das Páginas é uma atividade de construção que faz uso pesado de ferramentas de automatização. O conteúdo definido na atividade de engenharia é

fundido com os projetos de arquitetura, navegação e interface para produzir páginas em *HTML*, *XML* e outras linguagens orientadas a processo (como *Java*). Integração com *middleware* (*CORBA*, *DCOM* ou *JavaBeans*) também é feita durante esta atividade.

A atividade de *Teste* tenta descobrir erros em *applets*, *scripts* e formulários; e ajuda a garantir que o aplicativo irá funcionar corretamente em diferentes ambientes (com diferentes *browsers*).

Cada incremento produzido como parte do processo é revisto durante a *Avaliação do Cliente.* Neste ponto mudanças são pedidas (extensões de escopo ocorrem). Estas mudanças serão integradas ao sistema no próximo ciclo do processo incremental.

Murugesan destaca alguns passos para a construção de um aplicativo para a *Web* de sucesso: (MUR00)

- Entender o funcionamento geral e operacional do ambiente dos sistemas;
- Identificar e especificar requisitos técnicos e não-técnicos;
- Desenvolver uma arquitetura apropriada;
- Tratar satisfatoriamente as limitações não-técnicas;
- Identificar sub-projetos ou sub-processos para implementar a arquitetura;
- Desenvolver e implementar os sub-projetos;
- Incorporar mecanismos apropriados e efetivos para gerenciar a evolução e as manutenções.

O autor também destaca os problemas que normalmente ocorrem:

- Inconsistência das informações;
- Manutenção normalmente utiliza muitos recursos e tempo;
- Falta de escalabilidade;
- Necessidade de um “visual” comum;
- No início, requisitos são muito vagos;
- Os requisitos irão mudar consideravelmente durante o desenvolvimento e especialmente quando o aplicativo já estiver em uso;
- A tecnologia também vem mudando muito rapidamente.

Pesquisas conduzidas por McDonald e Welland mostram que grande parte dos desenvolvedores para a Web não utilizam um processo de desenvolvimento bem definido e documentado. E dos que estão utilizando processos bem definidos e

documentados, apenas alguns poucos estão utilizando modelos padronizados pela indústria de software; os outros estão utilizando modelos criados dentro da própria organização.

As pesquisas também mostraram que na maioria dos casos o processo se foca na Implementação. Análise de Requisitos e Projeto normalmente estão combinadas no começo do projeto e as atividades de Teste são conduzidas em conjunto com a implementação, isto quando são feitas. [MCD01]

5. Formulação e Análise

Formulação e Análise de sistemas e aplicativos para a *Web* representam uma seqüência de atividades de engenharia que começam com a identificação das metas e objetivos do aplicativo, e terminam com o desenvolvimento de um modelo de análise ou especificação de requisitos para o sistema. A Formulação permite que o cliente e o desenvolvedor estabeleçam um conjunto comum de metas e objetivos para a construção do aplicativo. Ela também ajuda a identificar o escopo do trabalho de desenvolvimento e fornece meios de determinar o sucesso do projeto. Análise é uma atividade técnica que identifica dados, funcionalidades e requisitos comportamentais de um aplicativo. [PRE01]

5.1 Formulação

As seguintes questões devem ser feitas no primeiro passo da etapa de formulação:

- Qual o principal motivo para desenvolvermos este aplicativo?
- Por que este aplicativo é necessário?
- Quem vai usar este aplicativo?

A resposta para cada uma destas perguntas deve ser determinada de maneira bem sucinta e objetiva. Através delas são identificadas as metas. Há basicamente duas categorias de metas:

- *Metas de informação*. Indicam a intenção de fornecer conteúdo específico e/ou informação para o usuário
- *Metas de aplicativo*. Indicam a habilidade de executar tarefas do aplicativo.

Quando todas as metas de ambos os tipos forem identificadas, um perfil de usuário é desenvolvido. Este perfil captura "características relevantes dos usuários potenciais" incluindo suas experiências, conhecimentos, preferências, etc.

Quando todas as metas e perfis de usuários estiverem desenvolvidos, a atividade de formulação irá focar a declaração de escopo do aplicativo para a *Web*. Em muitos

casos, as metas desenvolvidas estão integradas com esta declaração de escopo. Também é importante, neste estágio, indicar os graus de integração esperados e restrições de conectividade. [PRE01]

5.2 Análise

Durante esta etapa da Engenharia para a *Web*, quatro diferentes tipos de análises são conduzidos:

- **Análise de Conteúdo:** todo o conteúdo a ser fornecido pelo aplicativo é identificado. Conteúdo inclui textos, gráficos e imagens, dados de áudio e vídeo.
- **Análise de Interação:** a maneira pela qual o usuário interage com o aplicativo é descrita em detalhes.
- **Análise Funcional:** os cenários de uso criados na análise de interação irão definir operações que irão ser utilizadas no aplicativo, que implicam outras funções de processamento. Todas as operações e funções são descritas em detalhe.
- **Análise de Configuração:** O ambiente e a infra-estrutura na qual o aplicado reside são descritos em detalhe. O aplicativo pode estar na Internet, em uma *Intranet* ou em uma *Extranet*.

Pressman afirma que: apesar da especificação detalhada de requisitos ser recomendada para aplicativos complexos, raramente ela é feita. Isto fica especialmente claro no resultado das pesquisas realizadas por McDonald e Welland. A argumentação mais comum é que a evolução contínua dos aplicativos para a *Web* torna os documentos de requisitos obsoletos antes mesmo de completar o desenvolvimento. Apesar disto ser verdade em alguns casos, é necessário definir ao menos um modelo de análise para servir de base para a atividade de projeto que vem adiante. Minimamente se deve rever as informações coletadas, modificá-las conforme necessário e organizá-las em um documento que pode ser passado aos projetistas. [PRE01] [MCD01]

6. Projeto

As características de curto prazo de desenvolvimento e rápida evolução de sistemas para a *Web* forçam os desenvolvedores a realizarem um projeto que resolva os problemas imediatos e que, ao mesmo tempo, crie uma arquitetura que comporte uma evolução rápida. O problema, obviamente, é que na tentativa de resolver apenas o problema imediato, acaba-se comprometendo a capacidade evolutiva do aplicativo. Este é o dilema do projetista. [PRE01]

Com o objetivo de fazer um projeto efetivo, o engenheiro deve se concentrar no reuso de quatro elementos técnicos:

- **Métodos e Princípios de Projeto:** Modularidade eficiente (alta coesão e baixo acoplamento) e outras heurísticas da construção de software devem ser utilizadas também para a *Web*. Pode-se utilizar inclusive os métodos de projetos para sistemas orientados a objetos, pois a hipermídia define "objetos" que interagem através de um protocolo de comunicação que é bem parecido com o utilizado na orientação a objetos. Além disso, há grande variedade de métodos para projeto de hipermídia.
- **Regras de Ouro** *(Golden Rules)***:** Sistemas para a *Web* já vêm sendo construídos há uma década. Neste tempo, os projetistas desenvolveram um conjunto de heurísticas que podem ser reaplicadas durante o projeto de novos aplicativos.
- **Padrões de Projetos** (*Design Patterns*)**:** São abordagens genéricas utilizadas para resolver problemas genéricos que podem ser adaptadas para resolverem uma grande variedade de problemas mais específicos.
- **Modelos** *(Templates)***:** um modelo pode ser utilizado para fornecer um esqueleto para qualquer tipo de padrão de projeto que será utilizado no aplicativo.

6.1 Projeto de Arquitetura

O Projeto de Arquitetura para sistemas para a *Web* tem foco na definição da estrutura hipermídia do aplicativo, na aplicação de padrões e na construção de modelos

(templates) para montar a estrutura e permitir reuso. Uma atividade paralela chamada de *Projeto de Conteúdo*, deriva a estrutura geral e o esboço detalhado do conteúdo que será apresentado no aplicativo.

Nesta etapa é definida a estrutura que será utilizada, ou seja, a maneira como o conteúdo será apresentado ao usuário, e como a navegação será realizada. Pressman mostra quatro tipos de estruturas que podem ser utilizadas: [PRE01]

- *Estrutura Linear*: utilizada quando há seqüência previsível de interações, e eventualmente alguma variação. Um bom exemplo seriam apresentações de tutoriais com várias páginas de informação, além de gráficos e vídeos relacionados. Neste caso o conteúdo é predominantemente linear.
- *Estrutura de Grade:* aplicada quando o conteúdo pode ser organizado categoricamente em duas (ou mais) dimensões. Um exemplo seria uma loja de instrumentos musicais, os produtos poderiam ser separados por tipo (violões, guitarras, contra-baixos, etc.) ou fabricantes, e o usuário teria a opção de escolher como quer navegar.
- *Estrutura Hierárquica:* a mais comum. Nela o usuário pode navegar por toda a hierarquia, não apenas na vertical, mas também na horizontal. Isto é feito através de *links* que levam a outra parte da estrutura. É uma estrutura que permite navegação rápida, mas que pode confundir o usuário.
- *Estrutura de Rede ou "Pure Web":* similar ao modo como funciona a arquitetura de sistemas orientados a objetos. Nela cada componente (neste caso páginas) são projetados de modo que possam passar comandos (via *links* de hipertexto) para virtualmente qualquer outro componente do sistema. Esta abordagem cria bastante flexibilidade de navegação, mas pode confundir o usuário.

Estes modelos de arquitetura descritos podem ser combinados para formar estruturas compostas. Por exemplo, a estrutura pode ser predominantemente hierárquica, mas uma parte dela pode ter características lineares, e uma outra parte ter uma estrutura de rede. O objetivo é criar a estrutura ideal para o conteúdo a ser apresentado.

6.2 Projeto de Navegação

Uma vez que a arquitetura está criada e os componentes (páginas, *scripts*, *applets*, etc.) já foram identificados, é hora do projetista definir caminhos que permitam ao usuário ter acesso aos conteúdos e aos serviços. Para tanto o projetista deve: identificar as semânticas de navegação para diferentes usuários e definir os mecanismos para realizar a navegação.

Um aplicativo complexo normalmente tem vários tipos de usuários. Por exemplo: visitantes, usuários registrados, etc. Cada tipo de usuário pode ser associado com diferentes níveis de acesso a conteúdo e diferentes serviços. Um visitante pode ter acesso apenas a um conteúdo limitado, enquanto que um usuário registrado terá acesso a uma quantidade muito maior de conteúdo.

O objetivo desta fase é criar uma unidade semântica de navegação *(semantic navigation unit – SNU)* para cada objetivo associado a cada tipo de usuário. A estrutura do *SNU* é composta de um conjunto de sub-estruturas navegáveis que podemos chamar de caminhos *(ways of navigating – WoN)*. Cada um desses caminhos representará a melhor maneira de navegar para que um determinado usuário atinja sua meta ou sub-meta. A estrutura de um caminho (*WoN)* é feita de um conjunto de nós relevantes *(navigational nodes – NN)* conectados por *links (navigational links)*, incluindo algumas vezes outras *SNUs*.

A próxima etapa é escolher como os *links* serão identificados. Dentre as opções estão: textos, ícones, botões, etc. O projetista deve escolher o que achar mais apropriado para o conteúdo e consistente com as heurísticas que levam a uma boa interface.

6.3 Projeto de Interface

Os métodos para construção de interfaces utilizados na Engenharia de Software podem ser aplicados também para a *Web*, porém as características dos aplicativos para a *Web* requerem algumas considerações adicionais. [PRE01]

Na *Web* a interface tem um papel ainda mais importante que no software tradicional, ela é a primeira impressão. Uma interface mal desenhada pode desapontar o usuário e ele pode procurar outro *site*. E ainda pior, provavelmente ele não voltará mais no *site* que ele não gostou. Nielsen apresenta algumas recomendações simples que podem ser seguidas para construir uma boa interface: [NIE00]

- Erros no servidor, mesmo os menores, podem fazer com que um usuário deixe o *site* e procure a informação ou serviço que deseja em outro lugar;
- Não se deve forçar o usuário a ler grandes quantidades de texto, principalmente se for texto explicando como operar o aplicativo ou navegar por ele;
- Avisos de "Em Construção" devem ser evitados, são *links* desnecessários que causam uma expectativa do usuário que com certeza irá se desapontar;
- Usuários não gostam de rolar a tela, informações importantes devem ser colocadas no topo, de forma que apareça logo que a página é carregada;
- Menus e barras de navegação devem ser projetados de forma consistente, e devem estar disponíveis em todas as páginas que o usuário irá navegar. Não se deve contar com as funcionalidades do *browser*;
- Opções de navegação devem ser óbvias, mesmo para o usuário casual. O usuário não pode ficar procurando pela tela até encontrar o que deseja.

O Projeto de Interface deve se preocupar bastante com a usabilidade. Muitas vezes o projetista quer criar algo complexo, utilizando todas as mais recentes inovações tecnológicas e acaba criando algo difícil de usar, que fará com que o usuário cometa muitos erros e desista de usar o *site*. Murugesan apresenta uma série informações e recomendações que devem ser levadas em consideração pelo desenvolvedor que quer criar um bom aplicativo para a *Web*: [MUR00]

- Pessoas gastam horas clicando em um *site* para encontrar uma simples informação.
- "Surfar" na *Web* é muito mais difícil do que parece.
- Os usuários (potenciais clientes) estão conseguindo navegar no seu site?
- Por que os humanos têm que se adaptar a tecnologia? Por que a tecnologia não se adapta aos humanos?
- Sua empresa não é o público-alvo.
- Você deve escrever seu *site* em uma linguagem que seus usuários compreendam.

- Grandes erros continuam acontecendo na *Web*: *links* quebrados, cores não padronizadas, *URLs* que mudam e fazem com que *links* externos para seu *site* fiquem quebrados.
- Na Internet, sobrevive o mais fácil de usar.
- Se o usuário não consegue achar o produto, ele não irá comprá-lo.
- Atenção com a usabilidade aumenta a porcentagem de visitantes que se tornam clientes.

O usuário na *Web* tem pouca paciência com *sites* lentos ou difíceis de navegar, eles não querem ter de aprender como usar um *site*, eles devem simplesmente conseguir utilizá-lo, caso contrário irão procurar a informação que desejam em outro lugar. Problemas de usabilidade ainda afetam a grande maioria dos *sites*, mesmo os *sites* mais conhecidos e já consolidados apresentam problemas que podem afastar visitantes que estão fazendo sua primeira visita. [MUR00] [BRE01]

7. Testes

Assim como na Engenharia de Software, na Engenharia para a *Web* as atividades de teste também visam encontrar erros. De fato, os testes nos aplicativos para a *Web* são um desafio ainda maior, pois estes aplicativos podem ser acessados utilizando diferentes *browsers*, sistemas operacionais, plataformas de hardware, etc.

Pressman apresenta uma abordagem que adota os princípios básicos para o teste de qualquer software e aplica estratégias e táticas que são recomendadas para sistemas orientados a objetos: [PRE01]

1. **O modelo de conteúdo é revisto para descobrir erros.** Esta atividade de teste é similar em muitos aspectos com a revisão de documentos impressos. Um *site* grande pode utilizar os serviços de um editor profissional que irá descobrir erros de tipografia e gramática, consistência do conteúdo, representações gráficas, dentre outros.
2. **O modelo de projeto é revisto para descobrir erros de navegação.** Cada cenário é exercitado de acordo com o projeto de arquitetura e navegação. Isto serve para encontrar erros de navegação onde o usuário não consegue chegar ao nó desejado. Além disso, cada *link* é testado para garantir que correspondem ao que foi especificado na *SNU* para cada tipo de usuário.
3. **Componentes selecionados passam por um processo de teste de unidade.** Nos aplicativos para a *Web* o conceito de unidade muda. Cada página contém conteúdo, *links*, *forms*, *scripts*, etc. Nem sempre é possível testar cada uma dessas características individualmente. Em muitos casos, a menor unidade testável é a página. No software tradicional o teste de unidade é focado em detalhes de algoritmo de um módulo e dos dados que fluem pela interface do módulo. Nos aplicativos para a *Web* este teste é focado pelo conteúdo, processamento e *links* que estão nas páginas.
4. **A arquitetura é construída e testes de integração são conduzidos.** A estratégia para teste de integração depende da arquitetura que foi escolhida. Se foi utilizada uma arquitetura linear, de grade ou hierárquica é possível integrar as páginas da mesma maneira que fazemos com

software tradicional. Porém, se foram utilizadas arquiteturas combinadas ou estrutura de rede, o teste de integração passa a ser similar a abordagem usada para orientação a objetos.

5. **O aplicativo já integrado é testado em sua funcionalidade geral e conteúdo fornecido.** Assim como na validação de software convencional, a validação de sistemas para a *Web* é focada nas ações do usuário e nas saídas do sistema para o mesmo. Para ajudar na construção de testes de validação o testador deve se basear em casos de uso (*use-case*).
6. **O aplicativo é implementado em diferentes configurações de ambientes e testado em sua compatibilidade com cada configuração.** São definidos todos os prováveis sistemas operacionais, *browsers*, plataformas de hardware e protocolos de comunicação. Testes são conduzidos para descobrir erros associados com cada uma das possíveis configurações.
7. **O aplicativo é testado por uma população controlada de usuários.** São selecionados de usuários que representem cada tipo de usuário que o sistema terá. O aplicativo é testado por estes usuários e os resultados de suas interações são avaliados para encontrar erros de conteúdo e navegação, questões de usabilidade e compatibilidade, bem como desempenho e confiabilidade do aplicativo.

8. Gerenciamento

Por causa do curto período de desenvolvimento e da constante evolução de aplicações para a *Web*, muitos perguntam: "É realmente necessário gastar tempo gerenciando o desenvolvimento?" Muitos desenvolvedores acabam optando por pouco ou nenhum gerenciamento. Isto não quer dizer que eles estejam certos. [PRE01]

O desenvolvimento para a *Web* é complicado. Muitas pessoas são envolvidas, freqüentemente trabalhando em paralelo. A combinação de tarefas técnicas e não técnicas que ocorrem são um desafio para qualquer grupo de profissionais. Para evitar confusões, frustrações e falhas, um planejamento precisa ocorrer, riscos precisam ser considerados, um cronograma precisa ser estabelecido e acompanhado, e mecanismos de controle precisam ser definidos. A este conjunto de atividades chamamos "Gerenciamento".

8.1 A equipe de desenvolvimento

A criação de um aplicativo de sucesso para a *Web* demanda um grande conjunto de habilidades, pois são muitos aspectos que devem ser considerados. A equipe pode ser organizada quase da mesma forma que no desenvolvimento de software tradicional, porém as tarefas serão bastante distintas.

Os seguintes papéis podem ser distribuídos entre os membros da equipe: [PRE01]

- **Desenvolvedores e Provedores de Conteúdo**. Este pessoal irá se concentrar na coleta e geração de conteúdo. Podem vir de diversas áreas não relacionadas com software. Por exemplo, o pessoal de marketing ou vendas poderá fornecer informações e imagens de produtos, os criadores de mídia poderão fornecer material de áudio e vídeo, projetistas gráficos poderão fazer o projeto de layout e conteúdo estético, etc.
- ***Web Publisher***. O vasto conteúdo gerado pelos desenvolvedores de conteúdo precisa ser organizado para inclusão no aplicativo. Além disso, alguém precisa fazer a conexão entre o pessoal técnico que constrói o aplicativo e o pessoal não-técnico que desenvolve o conteúdo. Este papel

é realizado pelo *Web Publisher*, que precisa entender o conteúdo, bem como a tecnologia do aplicativo.

- ***Web Engineer.*** Este se envolve em várias atividades durante o desenvolvimento, incluindo levantamento de requisitos; análise; projeto arquitetural, de navegação e de interface; implementação; e teste. Ele também precisa ter um conhecimento sólido de tecnologia de componentes, arquitetura cliente/servidor, *HTML*/*XML*, tecnologias de banco de dados, e também conhecimento de conceitos multimídia, plataformas de hardware e software e segurança de redes.
- **Especialista de Suporte.** Este papel é designado para as pessoas que serão responsáveis por dar continuidade no suporte ao aplicativo para a *Web*. Como este tipo de aplicativo evolui continuamente, este especialista é responsável pelas correções, adaptações e melhorias no *site*, incluindo atualização de conteúdo, implementação de novos procedimentos e mudanças na navegação.
- **Administrador**. Também conhecido como *Web Master*, esta pessoa é responsável pela operação do *site* no dia-a-dia, incluindo desenvolvimento e implementação de regras para operação do aplicativo; fixação de procedimentos de suporte; implementação de procedimentos de segurança e direitos de acesso; medição e análise de tráfego; coordenação de mudanças de procedimentos; coordenação de especialistas de suporte.

McDonald e Welland destacam que as equipes de desenvolvimento para aplicativos *Web* são normalmente menores que as equipes de desenvolvimento de software tradicional. Em ambos os casos as equipes são gerenciadas em pequenos grupos, mas a semelhança acaba aí, pois no desenvolvimento de software tradicional as equipes são divididas em unidades menores para resolver diferentes problemas e executar diferentes tarefas. Mas no desenvolvimento para a *Web*, as equipes são divididas em grupos multidisciplinares, que construirão diferentes seções do aplicativo para a *Web*, mas em geral irão trabalhar em problemas similares. No decorrer do desenvolvimento de um software tradicional, as equipes devem interagir entre si, isto normalmente é feito através de interfaces pré-definidas, com cada equipe vendo o trabalho das outras equipes

como caixas-pretas Na engenharia para a *Web* as equipes precisam se comunicar ainda mais, com o objetivo de reduzir o esforço e garantir consistência. [MCD01]

8.2 Gerenciamento de Projeto

Na teoria, a maioria das atividades de gerenciamento de projeto utilizadas na Engenharia de Software pode ser utilizada também na Engenharia para a *Web*. Mas na prática, a abordagem é consideravelmente diferente.

O desenvolvimento de aplicativos para a *Web* é uma área relativamente nova e há poucos dados históricos que podem ser utilizados para fazer estimativa. Até agora, nenhum tipo de métrica foi publicado e ainda há pouca discussão de como devem ser estas métricas. Com isso, estimativas são baseadas apenas em experiências com projetos similares. Mas quase todo aplicativo para a *Web* quer inovar em alguma coisa, oferecendo algo novo e diferente. Isto acaba fazendo com que estimativas baseadas em experiência com outros projetos, apesar de úteis, estejam sujeitas a uma alta margem de erro.

Grande parte dos aplicativos na *Web* é construída por terceiros, especializados neste tipo de desenvolvimento. Neste caso, é útil para a empresa contratante fazer algumas tarefas antes de procurar alguém para fazer o trabalho: [PRE01]

1. Muitas das atividades de análise devem ser feitas internamente, incluindo a definição do público-alvo; dos objetivos; das informações e serviços a serem fornecidos; e das medidas quantitativas e qualitativas que serão utilizadas para medir o sucesso. Tudo isto deve ser documentado na especificação do produto.
2. Um esboço do projeto deve ser criado, pois economizará tempo e custo para o desenvolvedor, que terá uma idéia melhor de como deverá ser o aplicativo. Estas informações também devem ser adicionadas à especificação do produto.
3. Um esboço do cronograma deve ser definido e acompanhado.
4. Os níveis de interação entre o contratante e o contratado devem ser identificados, incluindo as responsabilidades de cada um.

8.3 Gerenciamento de Configuração

As estratégias utilizadas na Engenharia de Software são aplicáveis, porém táticas e ferramentas devem ser adaptadas para as características dos aplicativos para a *Web*. Quatro limitações devem ser consideradas quando desenvolvendo táticas para o gerenciamento de configuração: [PRE01]

- **Conteúdo.** Um aplicativo para a *Web* típico possui bastante conteúdo – textos, gráficos, *applets*, arquivos de áudio e vídeo, formulários, tabelas, etc. O desafio é organizar todo este conteúdo em um conjunto racional de objetos e então estabelecer mecanismos de controle de configuração apropriados para estes objetos.
- **Pessoas.** Como o desenvolvimento do aplicativo para a *Web* é contínuo, qualquer pessoa pode criar conteúdo. Muitas delas não têm conhecimentos em engenharia de software e desconhecem as necessidades de gerenciamento de configuração. Estes aplicativos acabam crescendo de forma descontrolada.
- **Escalabilidade.** As técnicas e controles aplicados a aplicativos para a *Web* pequenos não são bem escaláveis. É comum ver aplicativos crescendo em tamanho e complexidade, e com isso pequenas mudanças acabam tendo efeitos inesperados e problemáticos.
- **Política.** Quem é o dono do aplicativo? Esta questão é bastante discutida em grandes e pequenas companhias, e a resposta tem impacto significativo nas atividades de gerenciamento e controle.

Resumidamente, poderíamos dizer que o Gerenciamento de Configuração para a *Web* ainda está começando. O processo convencional não funciona como deveria. A maioria das técnicas existentes não conta com elementos que permitam adaptá-las para a *Web*. Estas limitações precisam ser contornadas antes que tenhamos um controle de configuração disponível par aplicativos *Web*.

Conclusão

A evolução rápida da *Web* e o impacto que ela tem causado nos últimos anos é bem significante. A maneira desordenada com que a maior parte dos aplicativos para a *Web* são construídos é preocupante, principalmente quando analisamos as pesquisas que mostram que a grande maioria dos *sites* tem problemas de funcionalidade e/ou usabilidade.

As técnicas criadas e já consolidadas na Engenharia de Software podem ser utilizadas e isto é uma vantagem. Porém, é preciso fazer algumas adaptações, tendo muito cuidado com todas as características específicas de sistemas para a *Web* que foram estudadas neste trabalho.

Infelizmente poucos desenvolvedores utilizam as técnicas de engenharia, e os poucos que utilizam o fazem de forma restrita ou errônea. Este é um problema que também ocorre na Engenharia de Software, mas na Engenharia para a *Web* a incidência é ainda maior. Os problemas ocorrem principalmente nas áreas de Análise, Requisitos, Testes, Validação e Manutenção, que são áreas importantes que ainda não recebem a atenção necessária, pois a maior parte dos desenvolvedores concentram-se apenas na fase de Implementação.

Os estudos mostram que o uso de técnicas de engenharia é eficaz e eficiente, portanto os esforços devem ser concentrados nesta adaptação das técnicas utilizadas na Engenharia de Software, incluindo o desenvolvimento de métricas e técnicas para o gerenciamento de projeto. Paralelamente precisa haver um trabalho para conscientizar desenvolvedores da importância do uso destas técnicas, mostrando como as mesmas poderiam tornar seu trabalho mais eficiente, trazendo também melhores resultados.

Referências Bibliográficas

[BIE98] BIEBER, Michael. "Web Engineering". New Jersey Institute of Technology, 1998. <http://www-ec.njit.edu/~bieber/web-engineering.html>

[BRE01] BREVE, Fabricio e WELLER, Daniel. "Métodos de Avaliação para Sites de Entretenimento". Universidade Metodista de Piracicaba, 2001.

[GIN01] GINIGE, Athula e MURUGESAN, San. "Web Engineering: An Introduction". IEEE Multimedia. Janeiro-Março 2001.

[GIN01b] GINIGE, Athula. "Engineering A Better Website". University Of Western Sydney. Austrália, Outubro de 2000. <http://aeims.uws.edu.au/talks/eng-web-sites.pdf>

[GIN01c] GINIGE, Athula. "Web Engineering in Action". University Of Western Sydney. Austrália, Outubro de 2000 <http://aeims.uws.edu.au/talks/Webe-in-action.pdf>

[MCD01] MCDONALD, Andrew e WELLAND, Ray. The University, Glasgow G12 8QQ, Scotland. 2001. <http://www.dcs.gla.ac.uk/~andrew/webe2001.pdf>

[MUR00] MURUGESAN, San. "Web Engineering For Sucessful Web Application Development". University Of Western Sydney. Austrália, Outubro de 2000. <http://aeims.uws.edu.au/Talks/Web-ApWeb2000.PDF>

[NIE00] NIELSEN, Jakob. "Designing Web Usability". New Riders Publishing, 2000.

[PRE01] PRESSMAN, Roger S. "Software Engineering: A Practioner's Approach". McGraw-Hill. 5ª edição. 2001.