Sistemas de Informação
Disciplina: Gestão de Sistemas Digitais na WEB
Professor: José Maurício S. Pinheiro

AULA 2 – Conceitos da Engenharia WEB

O número de sistemas e aplicativos para a Internet está em pleno crescimento e muda a forma como tratamos do desenvolvimento de sistemas computacionais. 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. Por esses motivos, é necessário o uso de metodologia para a construção destes sistemas. Os aplicativos para uso na Internet 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.

1. Introdução

Na década de 1950, um dos maiores problemas na criação de softwares era a falta de qualidade, pois o seu desenvolvimento acontecia sem nenhuma forma de estruturação e planejamento. Com o passar do tempo notou-se que quanto maior o sistema, mais complicada sua implementação e manutenção. Nesse contexto, vários problemas no desenvolvimento de software surgiram, tais como: atraso na entrega do produto final, insatisfação do cliente, falta de funcionalidades do produto final, dificuldade na manutenção do produto, entre outros problemas que aumentavam consideravelmente o custo do software. Esta fase que se estendeu pela década de 1960 ficou conhecida como "*The Software Crisis* (Crise do Software)" (Fig. 1).

Figura 1 - Crise do software

Para conter esta crise, em 1968 aconteceu a *Conference on Software Engineering* (Conferência sobre Engenharia de Software, da OTAN), onde foram discutidas soluções para os problemas observados, e então surgiu o termo "Engenharia de Software", termo proposto por Fritz Bauer.
No início da Internet, ou simplesmente Web, também aconteceu de as páginas e aplicativos serem desenvolvidos sem uma metodologia ou processo para apoiar seus criadores. No entanto, à medida que eles cresceram em complexidade,

tornou-se necessário aplicar métodos disciplinados de Engenharia de Software, adaptados para essa nova plataforma de implementação.
Características do ambiente Web, como heterogeneidade, adaptabilidade, concorrência, carga imprevisível, mobilidade, disponibilidade, sensibilidade ao contexto, evolução dinâmica, imediatismo, segurança e estética imprimiram uma nova dinâmica aos processos já existentes, dando origem a uma nova subárea da Engenharia de Software, a Engenharia Web (WebE), que estabelece e usa os princípios de engenharia e abordagens disciplinadas para o desenvolvimento, implantação e manutenção de aplicações baseadas na Web (Fig. 2).

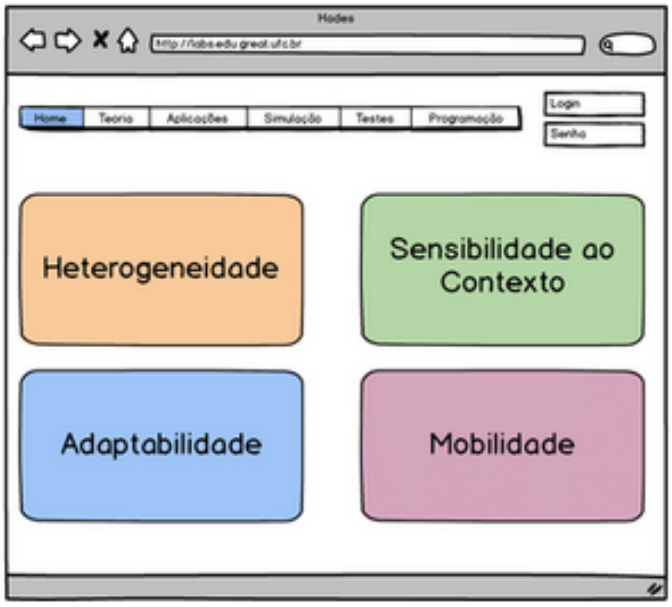

Figura 2 - Ambiente Web

2. Diferenças entre Engenharia de Software e Engenharia WEB

Uma das principais diferenças entre as duas Engenharias é o ciclo de vida do produto a ser desenvolvido. A evolução do Software tradicional (Fig. 3) é muitas vezes menor que a evolução de um aplicativo para WEB (Fig. 4), devido as tecnologias e informações estarem em constante evolução. O que influi realmente no ciclo de vida de uma aplicação para WEB é seu desenvolvimento ser uma mistura de arte e programação, entre marketing e computação, entre relações internas e externas além da multidisciplinaridade envolvida fazendo com que o seu desenvolvimento se torne um desafio.

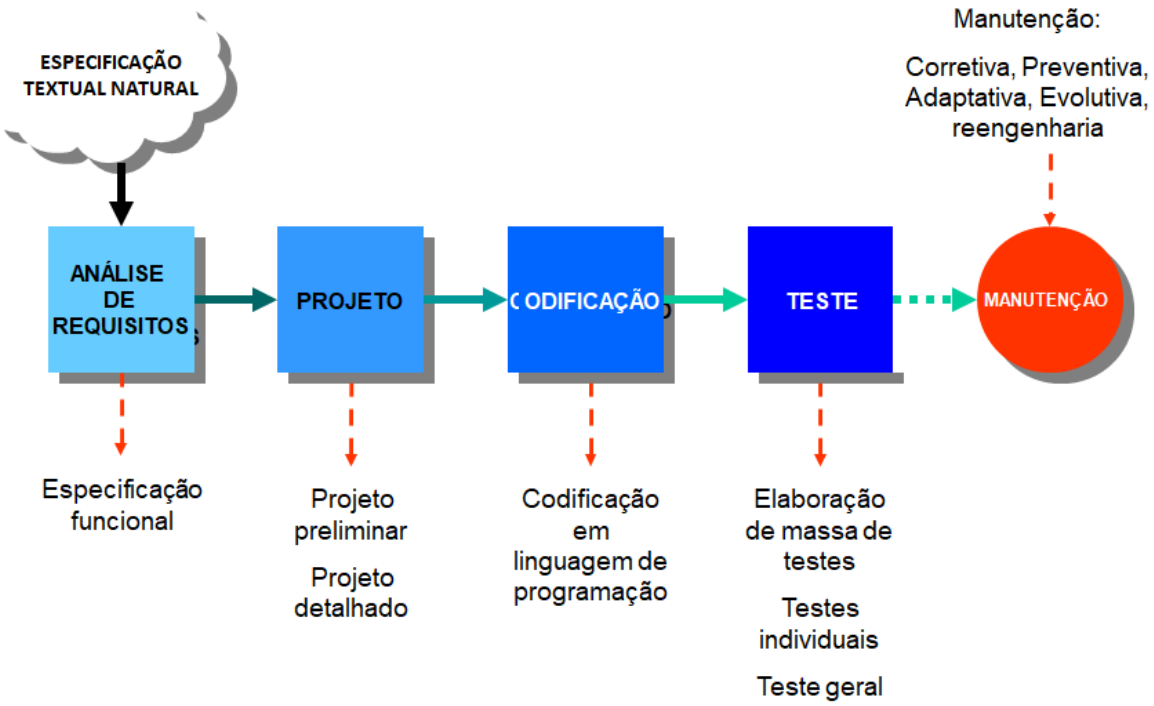

Figura 3 - Ciclo tradicional

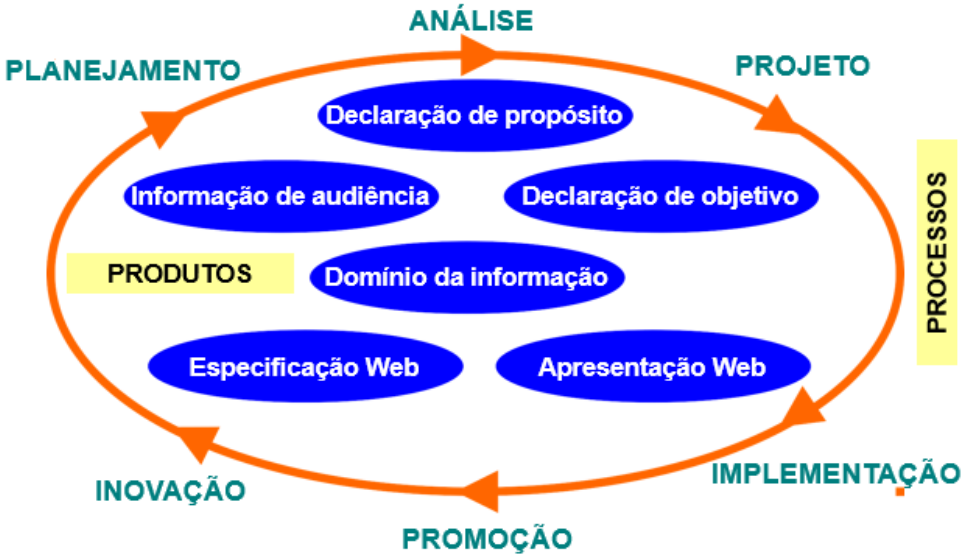

Figura 4 - Ciclo Web

Quanto ao Público alvo, uma aplicação convencional possui um público muitas vezes definido de usuários, que pode ser controlado, enquanto no desenvolvimento para WEB, além de diversificado, o número de usuários pode ser muito maior, às vezes sendo impossível saber exatamente o perfil de todos os usuários que irão interagir com o sistema a ser construído.

As tecnologias utilizadas na produção de software tradicional aplicando técnicas de engenharia de Software são mais estáveis, isto decorre do fato que o desenvolvimento deste tipo de software ocorre a muito mais tempo, além disso, existem excelentes ferramentas disponíveis para análise, projeto, implementação e testes. Já no desenvolvimento de WEB as tecnologias estão em constante

evolução, embora existam diversas ferramentas ao nível de implementação, existem poucas para análise, projetos e outras atividades para o desenvolvimento.

3. Websites e WebApps

Existem diversos tipos de website. Há páginas que são estritamente informativas, como páginas educacionais, algumas com grande volume de elementos multimídia, como os sites de museus e enciclopédias. Outras, no entanto, são sistemas de informação baseados na Web, como é o caso de lojas virtuais (e-commerce), ambientes cooperativos, sistemas de gerência empresarial, dentre outros. Esse tipo de website é conhecido como Sistemas de Informação Web (Web-based Information System - WIS) (Fig. 5), e ao conjunto amplo de todas as aplicações Web como WebApps (abreviação de Web Applications).
Sobre de WIS, tecnologias para implementação deste tipo de aplicação Web se desenvolvem rapidamente. Em 1994, com o advento do Common Gateway Interface (CGI), programas em linguagens como C ou PERL poderiam ser ativados por requisições de navegadores Web, fazendo com que um servidor Web pudesse retornar ao cliente o que fosse impresso pelo programa em sua saída padrão. Hoje, existem containers que gerenciam automaticamente componentes criados em linguagens orientadas a objetos, provendo uma série de serviços automáticos, como gerenciamento de persistência, controle de transações, etc.

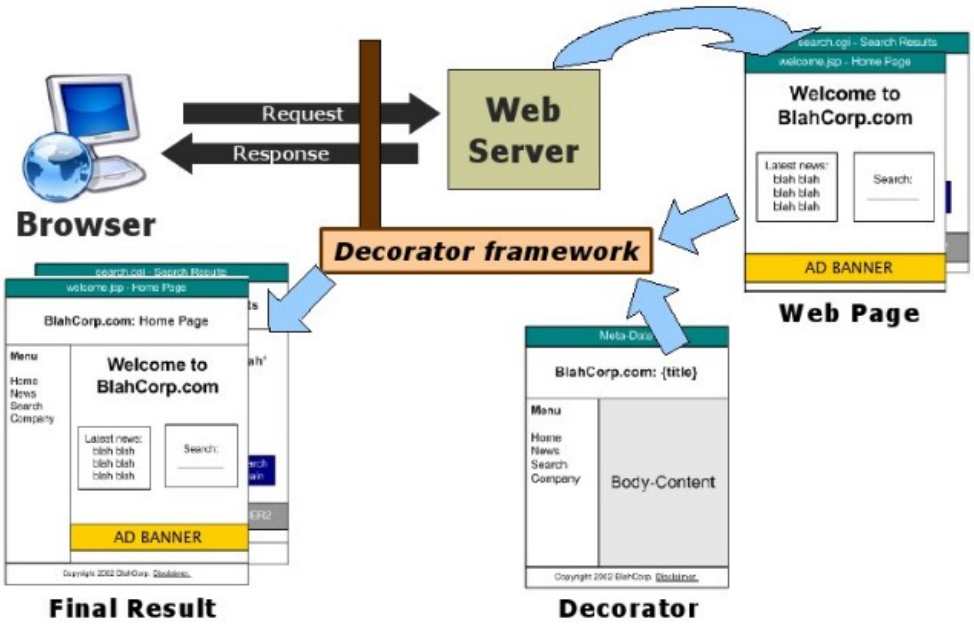

Figura 5 – Exemplo de arquitetura com aplicação Webapp

Destaca-se o surgimento de diversos frameworks que proveem uma boa infraestrutura para o desenvolvimento de WIS. Um Framework tem como principal objetivo resolver problemas recorrentes com uma abordagem genérica, permitindo ao desenvolvedor focar seus esforços na resolução do problema em si, e não ficar reescrevendo software. Trata-se de um conjunto de bibliotecas ou componentes que são usados para criar uma base onde sua aplicação será construída.
O uso desses frameworks, com o passar do tempo, tornou-se estado-da-prática e até mesmo influenciou na definição de padrões para desenvolvimento de aplicações distribuídas. Seu uso (ou reuso) auxilia a equipe de desenvolvimento a construir software mais rapidamente (vários componentes já estão prontos) e com

maior qualidade (os frameworks já foram extensivamente testados por seus criadores).
Alguns critérios que devem ser definidos para a criação de WebApp de qualidade:

- **Requisitos evoluem com o tempo:** Quando se inicia um projeto de WebApp, pode haver incerteza sobre alguns elementos das estratégias de negócios, do conteúdo e da funcionalidade a ser entregue, questões de interoperabilidade e outros aspectos de usabilidade;
- **As mudanças ocorrerão com frequência:** Como a incerteza é uma parte inerente da maioria dos projetos da WebApp, as mudanças nos requisitos são comuns. Além disso, o feedback do usuário (baseado em uma avaliação dos incrementos entregues) e alterações nas condições de negócios podem ocasionar mudanças;
- **Linhas de tempo são curtas:** Isso alivia a criação de documentação de engenharia, mas não impede que a análise do problema, o projeto e os testes devam ser documentados com detalhes;
- **Suporte a atividades de Engenharia Web**: comunicação, planejamento, construção e implantação.

Outras características que a Engenharia Web deve possuir para atender as demandas que uma aplicação Web exige e que se diferenciam da Engenharia de Software:

- Devido ao curto ciclo de desenvolvimento e curtos prazos para entrega do produto final bem menores que aplicações desktop, a Engenharia Web deve ser capaz de lidar com esta pressão;
- Por ser ligada não somente ao desenvolvimento em si, mas também ser orientada ao conteúdo, a Engenharia Web deve preocupar-se também com o desenvolvimento dos dados e da informação bem como o relacionamento entre eles;
- As equipes devem ser divididas em partes menores assim como em um processo de desenvolvimento de software convencional. Entretanto, as equipes devem poder ir além dos seus pares de trabalho e além de seus projetos para evitar esforços redundantes e garantir consistência;
- Deve haver um foco maior na fase de análise e avaliação e nas fases de requisitos e testes;
- Deve ser independente de tecnologia e ferramentas devido à rápida e contínua evolução das mesmas.

É importante ressaltar que a Engenharia Web deve ter um foco maior na segurança e na qualidade. O fato de as aplicações estarem disponíveis na Internet faz com que ataques sejam mais frequentes e qualquer brecha de sistema pode ser fatal. A segurança está diretamente ligada à qualidade que depende da aplicação de uma engenharia e processo bem definidos.

4. Modelo de Processo para Aplicações WEB

Como os aplicativos para WEB possuem evolução contínua, isso força um projeto que resolva rapidamente o problema, e ao mesmo tempo defina uma arquitetura para a aplicação que tem a capacidade de evolução ao longo do tempo. À medida que os aplicativos para Web evoluem de estáticos para dinâmicos é necessário aplicar um gerenciamento eficaz.

Pressman também propõe um modelo que inicia pela formulação, seguido pelo planejamento, análise, projeto, testes, gerência de projeto e configuração (Fig.6).

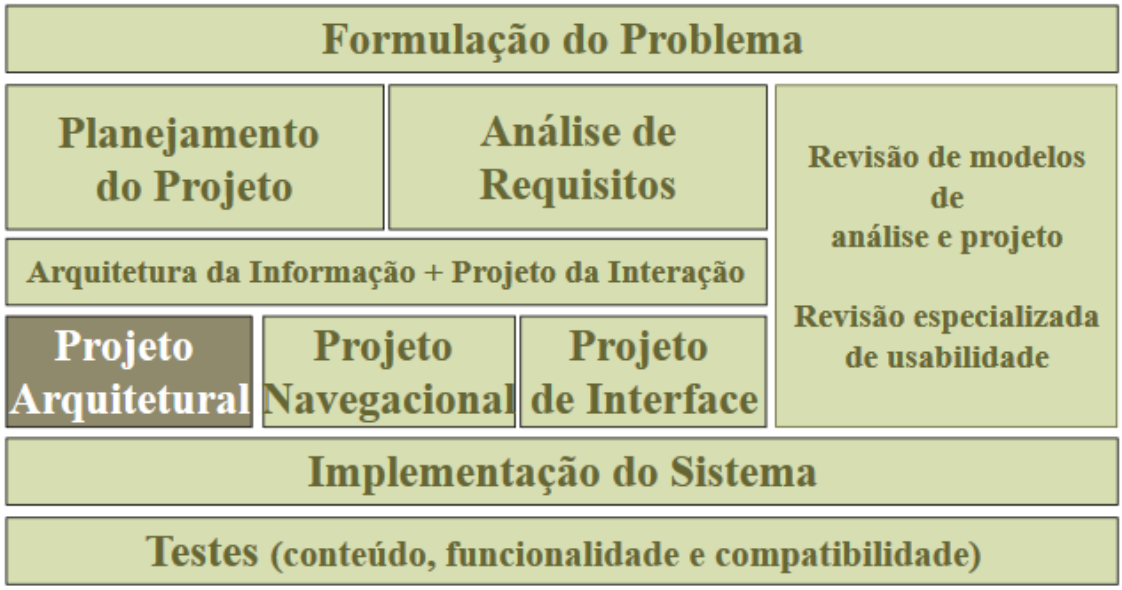

Figura 6 - Modelo de processo Web

4.1. Formulação e Análise

Formulação e Análise de sistemas e aplicativos para a Web representam uma sequência 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. Na análise, o conteúdo a ser oferecido pelo aplicativo será definido, assim como cada detalhe das funções, das operações, da configuração onde a aplicação reside e das maneiras como o usuário poderá interagir com o sistema.

4.1.1. Formulação

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

- Qual o principal motivo para desenvolver o aplicativo?
- Por que o 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 definido. 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.

4.1.2. Análise

Durante esta etapa da Engenharia 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 detalhes;
- **Análise de Configuração:** O ambiente e a infraestrutura na qual o aplicado reside são descritos em detalhes. O aplicativo pode estar na Internet, em uma Intranet ou em uma Extranet.

É necessário definir ao menos um modelo de análise para servir de base para a atividade de projeto. Minimamente se deve rever as informações coletadas, modificá-las conforme necessário e organizá-las em um documento que pode ser passado aos projetistas.

4.2. Planejamento do Projeto

O planejamento deve ser o próximo passo, nele o desenvolvedor deverá avaliar os riscos e custos associados ao desenvolvimento da aplicação. As características de curto prazo de desenvolvimento e rápida evolução dos sistemas 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 (Fig. 7).

Figura 7 - Planejamento do projeto

É necessário, portanto, desenvolver um modelo de desenvolvimento que atenda tais requisitos de forma eficiente. Existem quatro elementos técnicos definidos por Pressman que auxiliam o desenvolvedor da aplicação são eles: Modularidade eficiente, Regras de Ouro, Padrões de Projeto e Modelos:

- **Modularidade eficiente:** alta coesão e baixo acoplamento e outras heurísticas da construção de software devem ser utilizadas 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 vários anos. Assim, os projetistas desenvolveram um conjunto de heurísticas que podem ser reaplicadas durante o projeto de novos aplicativos. Uma lista criada em 1986, por Ben Shneiderman, professor da Universidade de Maryland, elenca 8 regras de ouro:

    1. **Esforço pela consistência** - As sequências consistentes de ações devem se repetir em situações semelhantes; as mesmas terminologias devem ser utilizadas em avisos, menus e telas de ajuda; consistência de cores, layout, capitalização e fontes devem ser empregada por toda parte. Exceções como a confirmação exigida do comando de exclusão ou repetição de senha devem ser compreensíveis e em número limitado.
    2. **Atender à usabilidade universal** - Reconhecer as necessidades dos usuários e projetar com flexibilidade, facilitando a transformação de conteúdo. Diferenças entre iniciantes e experientes, faixas etárias, incapacidades e diversidade tecnológica enriquecem o leque de requisitos que orientam o projeto. Inclusão de recursos para os novatos, como explicações, e recursos para especialistas, como atalhos, pode enriquecer o design da interface e melhorar a qualidade do sistema.

3. **Oferecer um feedback informativo** - Para cada ação do usuário, deve haver um feedback do sistema. Para ações frequentes e de menor importância, a resposta pode ser simples, enquanto para ações esporádicas e importantes, a resposta deve ser mais substancial. A apresentação visual dos objetos de interesse pode proporcionar um ambiente conveniente para mostrar as mudanças de forma explícita.
4. **Diálogos que indiquem o fim de uma ação** - Sequências de ações devem ser organizadas em grupos com um começo, meio e fim. Informação de feedback após a conclusão de um conjunto de ações dá aos usuários a satisfação de realização, uma sensação de alívio e uma indicação para se preparar para o próximo grupo de ações. Por exemplo, os sites de e-commerce movem os usuários da seleção de produtos para o checkout, terminando em uma página de confirmação clara que conclui a transação.
5. **Evitar erros** - Tanto quanto possível, projetar o sistema de tal forma que os usuários não possam cometer erros graves. Por exemplo, desabilitar com tons pouco visíveis os itens de menu que não são apropriados, e não permitir caracteres alfabéticos em campos numéricos. Se o usuário comete um erro, a interface deve detectar o erro e oferecer instruções simples, construtivas e específicas para recuperar a ação. Por exemplo, um usuário não deve ter que redigitar um formulário inteiro caso tenha inserido apenas o código postal inválido e deve ser orientado a reparar somente o dado incorreto. Os erros devem deixar o estado do sistema inalterado ou a interface deve dar instruções sobre como restaurar o estado.
6. **Permitir a fácil reversão de ações** - Tanto quanto possível, as ações devem ser reversíveis. Essa característica alivia a ansiedade, uma vez que o usuário sabe que os erros podem ser desfeitos e incentiva a exploração de opções desconhecidas. As unidades de reversão podem ser uma única ação, uma entrada de dado, ou um grupo completo de ações.
7. **Suportar o controle do usuário** - Usuários experientes querem ter a sensação de que estão no comando da interface e que ela responde às suas ações. Eles não querem surpresas no comportamento e ficam incomodados com sequências tediosas de entrada de dados, dificuldade na obtenção de informações importantes e incapacidade de produzir o resultado esperado.
8. **Reduzir a carga de memória de curta duração** - A limitação dos seres humanos para o processamento de informações na memória de curta duração (a regra de ouro é que podemos nos lembrar de aproximadamente sete pedaços de informação) exige que os designers evitem criar interfaces em que os usuários precisem memorizar informações de uma tela e, em seguida, usá-las em outra tela.

- **Padrões de Projetos (Design Patterns):** São abordagens genéricas utilizadas para resolver problemas diversos que podem ser adaptadas para resolverem uma grande variedade de problemas mais específicos (Fig. 8);

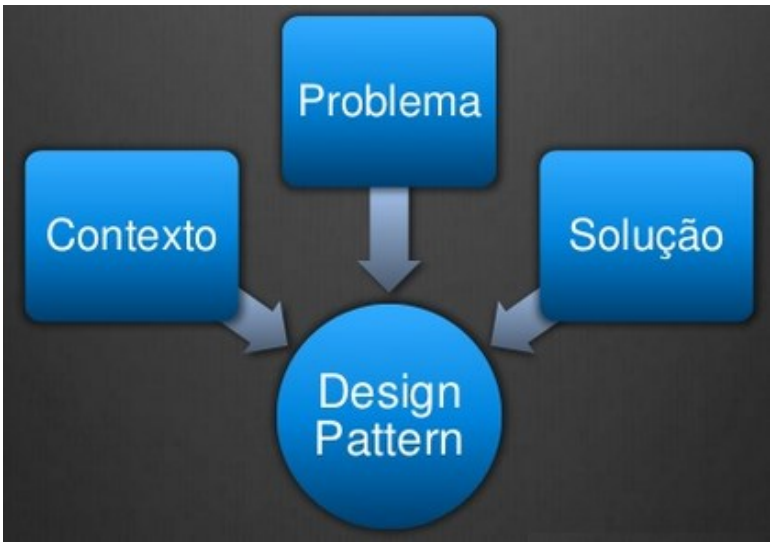

Figura 8 - Design pattern

- **Modelos (Templates):** um modelo pode ser utilizado para fornecer um esqueleto para qualquer tipo de padrão de projeto que será utilizado no aplicativo.

Seguindo as atividades propostas pelo modelo de processo, estão o projeto de arquitetura e projeto de navegação e o projeto de interface.

4.2.1. Projeto de Arquitetura

O Projeto de Arquitetura para sistemas 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:

- **Estrutura Linear -** o usuário é levado a seguir uma sequência de caminhos pré-estabelecidos pelo desenvolvedor. Um bom exemplo são as apresentações de tutoriais com várias páginas de informação, além de gráficos e vídeos relacionados. Neste caso, o conteúdo é predominantemente linear;
- **Estrutura de Grade** - permite dividir o conteúdo em categorias, essa estrutura facilita muito o entendimento do site e fornece opções de navegação ao usuário. Um exemplo é uma loja de instrumentos musicais, onde os produtos podem ser separados por tipo (violões, guitarras, contrabaixos etc.) ou fabricantes, e o usuário tem a opção de escolher como quer navegar;
- **Estrutura Hierárquica** - é a mais usada; apesar de tornar a navegação mais rápida, pode confundir o usuário, pois possibilita através de links deslocar a qualquer parte do site;

- **Estrutura de Rede ou "Pure Web"** - é bem flexível e 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.

As estruturas podem ser combinadas 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.

4.2.2. Projeto de Navegação

Uma vez que a arquitetura está criada e os componentes (páginas, scripts, applets etc.) já foram identificados, é hora de 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.
No projeto de navegação é definida a estrutura que será utilizada para apresentar o conteúdo ao usuário e como a navegação será realizada, ou seja, os caminhos e mecanismos de navegação para atingir os objetivos desejados pelo usuário. Um aplicativo complexo pode ter vários tipos de usuários. Por exemplo: visitantes, usuários registrados etc. Cada tipo de usuário pode ser associado a diferentes níveis de acesso a conteúdo e diferentes serviços. Um visitante pode ter acesso apenas a um conteúdo limitado, enquanto um usuário registrado terá acesso a uma quantidade de conteúdo exclusiva.
O objetivo 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 subestruturas 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 submeta. A estrutura de um caminho (WoN) é feita de um conjunto de nós relevantes (Navigational Nodes – NN) conectados por links (Navigational Links - NL), incluindo algumas vezes outras SNU's.

4.2.3. Projeto de Interface

No projeto de interface, a preocupação é voltada para a Interface, pois ela tem um papel fundamental: funciona como um cartão de visita do aplicativo WEB. A Interface deve ser bem projetada e planejada para atrair a atenção do usuário, impedindo assim que ele procure outro local capaz de satisfazê-lo melhor.
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.
A utilização de métodos de Engenharia de Software pode ser de grande utilidade para se obter uma boa interface: projetar bem os menus e barras de navegação de forma que fiquem disponíveis em todas as páginas que o usuário navegar; não contar com as funcionalidades do browser; opções de navegação devem ser óbvias, informações importantes devem ser colocadas no topo e permitir que o usuário saia do site caso ocorram erros no servidor.
A usabilidade realmente define a sobrevivência de uma aplicação na WEB e sites que apresentam usabilidade aumentam a probabilidade dos visitantes se tornarem

clientes. A aplicação deve ser construída utilizando uma linguagem fácil que o usuário compreenda e consiga navegar por todo site encontrando o que deseja. Algumas recomendações simples que podem ser seguidas para construir uma boa interface:

- 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.

4.2.4. Testes

Para finalizar o modelo processo para aplicações WEB, são realizados os testes, com a intenção de encontrar erros. Portanto, os testes para WEB exigem um esforço maior, pois, os aplicativos podem ser acessados utilizando diferentes browsers, sistemas operacionais, plataformas de hardware, além de estarem disponíveis para um número não definido de usuários.
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:

- 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.
- 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.
- 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, formulários, scripts etc. e 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.

- 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.
- 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.
- 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.
- 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.

4.3. Documentação do Projeto

É grande a importância da documentação do projeto de sistemas Web. Ao projetar um sistema, o projetista precisará da documentação do projeto do sistema para qualquer manutenção futura, lembrando que a equipe de profissionais envolvidos em determinado projeto pode ser transitória ou sofrer alguma modificação durante a construção do projeto. Outra necessidade que já se pode visualizar é em relação às peculiaridades de diferentes projetos. A documentação do projeto deve conter:

- Informações gerenciais, tais como versão, responsáveis, histórico de alterações;
- Descrição geral do sistema;
- Lista das visões consideradas e informações acerca do mapeamento entre elas.

4.4. Gerenciamento

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”.
É importante observar que a gerência de projeto e a gerência de configuração devem ser aplicadas a todas as etapas anteriores, pois, possibilitam o controle tanto do conteúdo quanto do pessoal envolvido no desenvolvimento.

4.4.1. 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:

- 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;
- Um esboço do projeto deve ser criado, pois economizará tempo e custo para o desenvolvedor, que terá uma ideia melhor de como deverá ser o aplicativo. Estas informações também devem ser adicionadas à especificação do produto;
- Um esboço do cronograma deve ser definido e acompanhado;
- Os níveis de interação entre o contratante e o contratado devem ser identificados, incluindo as responsabilidades de cada um.

4.4.2. 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:

1. **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;
2. **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;
3. **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.

4. **Política:** Definir a propriedade do aplicativo. Esta questão é bastante discutida em grandes e pequenas empresas e a resposta tem impacto significativo nas atividades de gerenciamento e controle.

5. Questões

1. O que foi a crise do software?

2. Qual a principal diferença entre Engenharia de Software e Engenharia WEB?

3. O que é um Padrão de Projeto?

4. Quais arquiteturas podem ser usadas no projeto para sistemas Web?

5. Por que a documentação de projeto é importante?