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

AULA 4 – Qualidade e Testes em Webapp

As aplicações Web (Webapp) podem ser consideradas como um sistema distribuído, com arquitetura cliente-servidor multicamadas, tendo como características: grande número de usuários com acesso concorrente; execução em ambientes heterogêneos compostos por diferentes hardwares, conexões de rede, sistemas operacionais, servidores Web e navegadores diferentes; natureza heterogênea devido à variedade de componentes e tecnologias empregadas no seu desenvolvimento (diferentes linguagens, modelos, etc.).
O desenvolvimento de aplicações Web destaca-se pelo rápido crescimento de requisitos, conteúdo e funcionalidade, e pelas constantes alterações sofridas durante o ciclo de vida. O desenvolvimento de sistemas baseado em Web é uma atividade contínua e sem releases específicas como ocorre no desenvolvimento de software convencional.
Projetos de desenvolvimento de aplicações Web surgem em resposta a uma necessidade ou uma oportunidade de negócio, e acontecem dentro de parâmetros predefinidos de tempo, custo e escopo e aborda técnicas onde é assimilada a clareza e a fisionomia de um site onde o layout é criado e definido a estrutura da arquitetura e uma estrutura de navegação. O grau de atendimento a estes parâmetros representa um indicador da qualidade do processo adotado o que, consequentemente irá influenciar a qualidade do produto gerado como resultado e na satisfação do cliente.

1. Qualidade de Software

Diferentes conceitos expressam a qualidade de um produto de software, tendo como base visões diferenciadas conforme a função do indivíduo ou sua posição no processo de desenvolvimento.

- Qualidade é estar em conformidade com os requisitos dos clientes. Neste caso, qualidade se exprime a partir da satisfação de todas as necessidades expressas pelo cliente durante a análise de requisitos;
- Qualidade é antecipar e satisfazer os desejos dos clientes. Neste caso, observa-se que a partir de um produto existente o desenvolvedor procura a evolução do produto, acrescentando novas funcionalidades ou apropriando o produto com novas tecnologias, muitas vezes antes do cliente solicitar evolução dele;
- Qualidade é escrever tudo o que se deve fazer e fazer tudo o que foi escrito. Neste caso, o reflexo do desenvolvimento precisa assegurar que tudo o que foi documentado será realizado pelo desenvolvedor.

Pressman (1995) define a qualidade de software como:

> Concordância com os Requisitos Funcionais e de Desempenho claramente colocados, Padrões de Desenvolvimento

> explicitamente documentados e características implícitas que são esperadas de todo software profissionalmente desenvolvido.

Seguindo essa definição, temos:

- **Requisitos de software** - formam a base a partir da qual a qualidade é avaliada. Representam as necessidades na forma de funcionalidades solicitadas pelo cliente. Falta de concordância com os requisitos é falta de qualidade;
- **Padrões especificados** - definem um conjunto de critérios de desenvolvimento que orientam a maneira como o software é construído. Se o critério não for seguido, certamente haverá falta de qualidade;
- **Requisitos implícitos** - na maioria das vezes não são mencionados, também são conhecidos como requisitos não funcionais (por exemplo, a possibilidade de portabilidade para diferentes sistemas operacionais, a necessidade de segurança em um site de e-commerce). Se o software atende aos requisitos explícitos, mas falha nos requisitos implícitos, a qualidade é suspeita.

De acordo com a ABNT, qualidade pode ser definida como o grau no qual um conjunto de características inerentes satisfaz a requisitos. Neste contexto, uma característica é considerada uma propriedade diferenciadora e um requisito representa uma necessidade ou expectativa que é expressa, geralmente, de forma implícita ou obrigatória.
A qualidade de um produto pode ser avaliada considerando os seguintes aspectos:

- **Técnicos** - no âmbito do projeto, a qualidade está associada ao cumprimento das instruções de trabalho que regem o projeto, tais como: comunicação, cumprimento das metas, prazos e orçamento de cada pacote de trabalho, gerenciamento dos riscos, etc., conforme fora planejado;
- **Funcionais** - a solução deve agradar ao cliente, ser de fácil administração, robusta, confiável e estável;
- **Documentação** - deve ser completa, clara, de fácil consulta e aderente aos padrões impostos pelo cliente.

A qualidade atingida no produto desenvolvido depende diretamente da qualidade do processo que foi seguido para desenvolver o produto.

2. Desenvolvimento Web e Qualidade

O desenvolvimento de uma aplicação Web (e softwares em geral) que culmina em um programa executável e satisfatório em termos de qualidade, envolve um conjunto de atividades cuja execução deve ser planejada de acordo às restrições impostas. O processo de desenvolvimento de uma aplicação Web estabelece as atividades pertinentes ao desenvolvimento de um projeto de software, indicando quando e por quem precisam ser executadas de forma a atingir os objetivos de qualidade propostos.

Entretanto, problemas de qualidade surgem de diversas fontes e suas origens normalmente podem ser rastreadas desde uma falta de entendimento até uma falta de atendimento das necessidades dos usuários finais. Portanto, o primeiro mecanismo de garantia de qualidade para o desenvolvimento de Webapp é a atividade de comunicação voltada para as necessidades do usuário.
No contexto do desenvolvimento de aplicações para Web, algumas práticas são consideradas princípios a serem seguidos e aplicados com o objetivo de obter um produto de qualidade. Alguns desses princípios são os seguintes:

- **Modularidade** – Consiste na divisão de sistemas complexos em partes menores e mais simples com características desejáveis e bem definidas (Coesão e Acoplamento). Decomposição é o ato de dividir um problema original em subproblemas, recursivamente. Composição é o ato de juntar os elementos componentes de um problema até chegar ao sistema completo. A modularidade tem como objetivo contribuir no entendimento de um problema complexo utilizando como estratégia a sua decomposição em partes menores. Adicionalmente contribui na manutenção e reusabilidade do sistema;
- **Separação de Conceitos** – Separar conceitos permite trabalhar com aspectos individuais e diferentes de um mesmo problema. Esta separação facilita o entendimento, focando a atenção em certas características mais significativas, através de um processo de abstração;
- **Generalidade/Especialidade** – Soluções genéricas tendem a ser mais caras em termos de recursos e em tempo de desenvolvimento, ao contrário das soluções mais específicas. No processo de produção de software estas questões devem ser cuidadosamente analisadas;
- **Rigor e Formalidade** – O processo de software é uma atividade criativa tendendo naturalmente à imprecisão. O rigor é a abordagem que produz produtos mais confiáveis pelo controle das variáveis envolvidas. Formalidade implica que o processo esteja dirigido e avaliado por leis matemáticas;
- **Incrementabilidade** – Caracteriza o processo em modo passo-a-passo, incrementalmente e prevê que o objetivo desejado seja atingido por aproximações sucessivas. Esta abordagem pode ser muito útil quando os requisitos iniciais não foram todos obtidos antes do início do desenvolvimento da aplicação;
- **Antecipação de Mudanças** – Desenvolver o sistema visando possíveis evoluções do produto, de forma a tornar a sua manutenção e adaptabilidade mais fáceis e baratas. Quando o sistema é finalmente liberado, novos requisitos podem ser descobertos e requisitos antigos atualizados através do feedback do usuário.

Princípios são importantes para o sucesso no desenvolvimento de software, mas não são suficientes para obter produtos de qualidade. É preciso aliar métodos, metodologias e ferramentas apropriadas para se aplicar os princípios no processo de produção de software (Fig. 1).

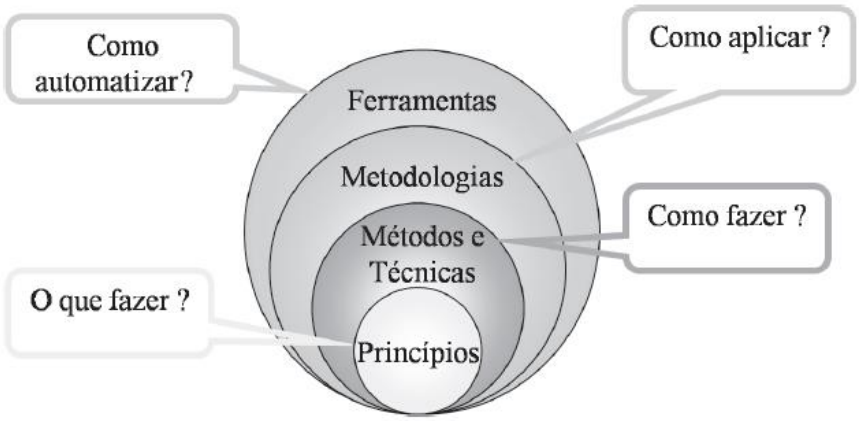

Figura 1 - Métodos, metodologias e ferramentas para desenvolver com qualidade

2.1. Características de Qualidade

As aplicações Web apresentam algumas características distintas quando comparadas às aplicações convencionais, tais como união de diferentes tecnologias em uma mesma aplicação, rápida evolução tecnológica, evolução constante de requisitos, acesso via browser e arquitetura diferenciada, possibilitando a execução de funções através de servidores ou do cliente.
A norma ISO/IEC 9126-1, define usabilidade como sendo "Um conjunto de atributos relacionados com o esforço necessário para o uso de um sistema interativo, e relacionados com a avaliação individual de tal uso, por um conjunto específico de usuários". Ela apresenta um conjunto genérico de características de qualidade para o software, que são confiabilidade, eficiência, funcionalidade, manutenibilidade, usabilidade e portabilidade (Fig. 2).

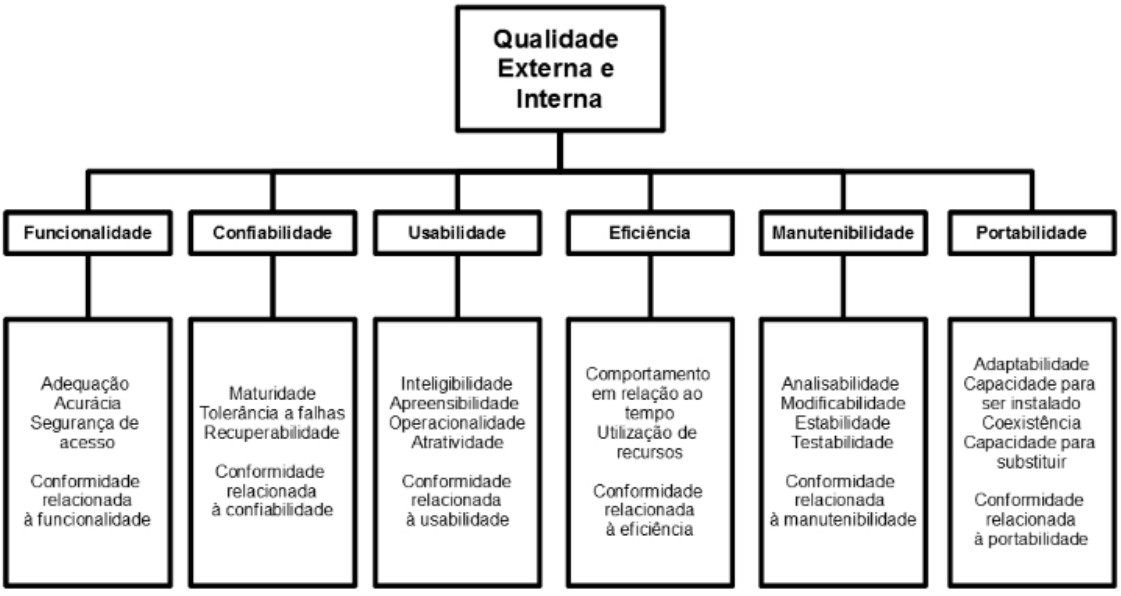

Figura 2 - Características e Subcaracterísticas de Qualidade (NBR ISO/IEC 9126-1)

Cada característica de qualidade é composta por subcaracterísticas que detalham um pouco mais cada requisito de qualidade a ser atingido em uma aplicação:

- **Funcionalidade -** capacidade do software em prover funções que atendam às necessidades quando é usado sobre determinadas condições.
  - Adequação - prover um conjunto apropriado de funções para tarefas e objetivos específicos do usuário;
  - Acurácia - prover resultados ou comportamentos corretos e de acordo com o nível de precisão necessário;
  - Interoperabilidade - interagir com um ou mais sistemas;
  - Segurança - proteger informações e dados que pessoas ou sistemas não autorizados possam ler ou modificá-los e negar acesso a eles;
  - Conformidade à funcionalidade - estar de acordo com normas ou convenções relacionadas à funcionalidade.

- **Confiabilidade** - Capacidade de o software em manter um nível especificado de desempenho quando usado sobre condições específicas.
  - Maturidade - evitar falhas como um resultado de um erro presente no software;
  - Tolerância a falhas - manter um nível especificado de desempenho no caso de erros ou por ter infringido a interface definida;
  - Recuperabilidade - restabelecer um nível especificado de desempenho e recuperar os dados diretamente afetados no caso de uma falha;
  - Conformidade à confiabilidade - estar de acordo com normas ou convenções relacionadas à confiabilidade.

- **Usabilidade** - capacidade do software de ser entendido, aprendido, utilizado, e atrativo para seus usuários, quando usado sobre condições específicas.
  - Facilidade de entendimento - possibilitar ao usuário o entendimento da aplicação, e como ela pode ser usada para uma tarefa ou condição de uso particular;
  - Facilidade de aprendizado - possibilitar ao usuário aprender como usar a aplicação;
  - Operabilidade - possibilitar ao usuário operar e controlar a aplicação;
  - Atratividade - ser atrativo ao usuário;
  - Conformidade à usabilidade - estar de acordo com normas ou convenções relacionadas à usabilidade.

- **Eficiência** - capacidade do software em prover desempenho apropriado, relativo à quantidade de recursos usados sob certas condições.
  - Comportamento de tempo - prover tempo apropriado de resposta e processamento;
  - Utilização de recursos - usar quantidade apropriada e tipos de recursos quando o software realiza suas funções sobre determinada condição;
  - Conformidade à eficiência - estar de acordo com normas ou convenções relacionadas à eficiência.

- **Manutenibilidade** - capacidade do software de ser modificado. Modificações podem incluir correções, melhorias ou adaptações do software a mudanças no ambiente e nos requisitos e especificações funcionais.
  - Analisabilidade - permitir diagnóstico de deficiências ou causas de falhas, ou identificação das partes que precisam ser modificadas;
  - Modificabilidade - possibilitar que uma modificação específica seja implementada;
  - Estabilidade - evitar efeitos não esperados após modificações;
  - Testabilidade - possibilitar que modificações sejam validadas;
  - Conformidade à manutenibilidade - estar de acordo com normas ou convenções relacionadas à manutenibilidade.

- **Portabilidade** - capacidade do software de ser transferido de um ambiente para outro.
  - Adaptabilidade - poder ser adaptado para diferentes ambientes sem aplicação de ações ou qualquer outro ajuste além daqueles já fornecidos pela aplicação;
  - Facilidades de instalação - poder ser instalado em um ambiente específico;
  - Coexistência - capacidade de coexistir com outras aplicações em um ambiente comum com compartilhamento de recursos;
  - Facilidade de substituição - capacidade de ser usado no lugar de outro software com o mesmo propósito existente no mesmo ambiente;
  - Conformidade à portabilidade - estar de acordo com normas ou convenções relacionadas à portabilidade.

A Tabela 1, apresenta o mapeamento das características de qualidade de acordo com a norma ISO/IEC 9126 (2002), destacando apenas a segurança, que representa uma subcaracterística de funcionalidade, devido a sua importância para as aplicações Web.

Tabela 1 - Mapeamento das Características de Qualidade

Característica ISO 9126	Característica necessária ou presente em aplicações Web
Funcionalidade	Atendimento aos requisitos funcionais solicitados pelos usuários. Precisão dos resultados e do comportamento da aplicação de acordo com os requisitos funcionais e não funcionais. ** **Sub-característica SEGURANÇA**: Acesso às funcionalidades somente por pessoas autorizadas. Evitar "ataques" diretos e indiretos ao servidor de aplicação e de banco de dados. Evitar acessos diretos às páginas dinâmicas (informando valores através de *links*)
Confiabilidade	Processamento correto de *links* (fluxo de navegação). Integridade dos *links* (evitar páginas inacessíveis, *links* quebrados ou sem ponteramento adequado). Apresentação de mensagens de erros, permitindo o retorno às funcionalidades. Processamento cancelado em caso de expiração da sessão.
Usabilidade	Preocupações com o nível de apresentação, tais como: Facilidade no uso da aplicação Aprendizagem das terminologias, menus e navegação através de diferentes páginas da Web
Eficiência	Tempo de resposta a uma solicitação. Tempo para carregar uma página. Carga de servidor de aplicação (quantidade de solicitações). Carga de servidor de banco de dados (quantidade de acesso).
Manutenibilidade	Não foi possível mapear atributos específicos para aplicação Web.
Portabilidade	Verificação de recursos específicos de *browser* e restrições para carregar uma determinada página (*scripts*, *plugin*, etc.). Adaptação a diferentes servidores, ou sistemas operacionais.

3. Usabilidade

Do ponto de vista do usuário, a interface é uma das partes mais importantes dos sistemas computacionais, porque por meio dela o usuário vê, ouve e sente. A usabilidade faz parte do contexto de desenvolvimento de Webapp e o termo faz parte do vocabulário técnico da Ciência da Computação, na área de Interação Humano Computador (IHC).
Usabilidade se refere à qualidade da interação entre sistemas e usuários e depende de vários aspectos, como a facilidade em aprender, a eficiência, a satisfação do usuário, entre outros e deve oferecer melhores aplicações web para os usuários, leigos ou não, auxiliando na criação de projetos que atendem as reais necessidades do usuário.
Para criar um aplicativo preparado para os desafios atuais é importante observar algumas situações:

- Softwares estão incorporados ao dia a dia do ser humano e o número de interessados na aquisição de sistemas aumenta constantemente;
- Usuários têm ideias diferentes de qual recurso é ou não necessário para um aplicativo, de forma que é necessário fazer um esforço para entender o problema antes de projetar a solução;
- Existe uma demanda maior por softwares mais complexos e inteligentes;
- Pessoas e empresas dependem de softwares, sendo que uma falha pode gerar desde pequenos inconvenientes até grandes problemas;

- Aplicativos precisam ser ajustados, modificados, corrigidos e melhorados, logo, eles precisam ser passíveis de manutenção.

Uma aplicação Web deve ser útil para o usuário, apresentar uma boa usabilidade através de um design de interação adequado. Uma boa experiência do usuário pressupõe que o sistema seja fácil de usar e que tenha credibilidade. O usuário deve sentir o desejo de usar a aplicação, que deve ser acessível e que propicia valor, de maneira que a experiência ao usar o sistema seja agradável e produtiva (Fig. 3).

Figura 3 - Usabilidade e experiência do usuário

A avaliação da usabilidade de interfaces visa verificar se elas atendem aos requisitos do usuário de forma que as funcionalidades do sistema sejam realizadas de modo efetivo, de forma eficiente, e que satisfaça as expectativas do usuário. Tendo em vista que o ciclo de desenvolvimento de software costuma ser longo, avaliações da usabilidade de diferentes versões de interfaces devem ser realizadas no decorrer do processo, como forma de minimizar erros e reduzir custos de produção do sistema.
Portanto, a usabilidade de um sistema é um conceito que se refere à qualidade da interação de sistemas com os usuários e depende de vários aspectos. Alguns destes fatores são:

- **Facilidade de aprendizado do sistema**: tempo e esforço necessários para que os usuários atinjam um determinado nível de desempenho;
- **Facilidade de uso:** avalia o esforço físico e cognitivo do usuário durante o processo de interação, medindo a velocidade de e o número de erros cometidos durante a execução de uma determinada tarefa;
- **Satisfação do usuário:** avalia se o usuário gosta e sente prazer em trabalhar com este sistema;
- **Flexibilidade:** avalia a possibilidade de o usuário acrescentar e modificar as funções e o ambiente iniciais do sistema. Assim, este fator mede também a capacidade de o usuário utilizar o sistema de maneira inteligente e criativa, realizando novas tarefas que não estavam previstas pelos desenvolvedores;

- **Produtividade**: se o uso do sistema permite ao usuário ser mais produtivo do que seria se não o utilizasse.

Muitas vezes, o desenvolvedor deve identificar quais destes fatores têm prioridade sobre quais outros, uma vez que dificilmente se consegue alcançar todos de forma equivalente. As decisões do projetista determinam a forma de interação entre usuários e sistemas. Frequentemente designers definem a facilidade de uso como sendo o aspecto de usabilidade prioritário e, por vezes, acabam desenvolvendo sistemas em que os usuários não cometem erros, mas também não têm muita opção de ação ou decisão. São os chamados de sistemas anti-idiotas (idiot-proof) e advogam que novas tecnologias serão mais eficazes quando projetadas para aumentar, ao invés de substituir, as capacidades dos usuários.
A usabilidade, de um modo geral, pode estar relacionada como um conjunto de fatores que qualificam o quão bem uma pessoa pode interagir com o sistema. Esses critérios podem ser:

- **Facilidade de aprendizado** (learnability) - se refere ao tempo e esforço necessário para que o usuário aprenda a utilizar o sistema com determinado nível de competência e desempenho. Qual a expectativa das pessoas ao se interagirem com um sistema computacional? Imagine um sistema de e-mail x software para declaração de imposto de renda. Qual o tempo e esforço de cada uma?
- **Facilidade de recordação** (memorability) - seres humanos esquecem determinados conhecimentos. Da mesma forma, ao reutilizar um determinado software, esse deve possui uma interface que auxilie o usuário a interagir com ele novamente.
- **Eficiência** (efficiency) - diz respeito ao tempo que o usuário leva para executar suas atividades. Assim, a maneira como a interface é criada é importante para o usuário executar suas tarefas com boa performance.
- **Segurança de uso** (safety) - todo usuário é passível de falhas. Desse modo, criar um sistema que possibilite uma maior segurança ao usuário em caso de falha é interessante. Como? A disposição de determinados botões é de extrema importância. Evitar colocar teclas como “apagar tudo” próximo de “gravar”, botões de refazer ou desfazer simples e fáceis de usar são exemplos

3.1. Satisfação do Usuário

Com o que está relacionado à satisfação do usuário? Imagine a seguinte situação: uma enfermeira plantonista tem que passar informações sobre o estado de seus pacientes no final de cada plantão em um software. Imagine agora que tenha as seguintes opções:

1. Enfermeira teve um plantão tranquilo e o software para passar os dados é de fácil uso;
2. Enfermeira teve um plantão tumultuado e tem dificuldades ao se interagir com o sistema.
3. Enfermeira teve um plantão tumultuado e o software é de fácil uso;

4. Enfermeira teve plantão tranquilo e o tem dificuldades ao se interagir com o software.

Qual o grau de satisfação dela em cada uma das seguintes situações? Imagine agora que a mesma enfermeira arruma emprego em um outro hospital e o procedimento para passar plantões é de um software similar. O que muda nesse contexto? Note que ela já possui uma experiência prévia. Assim a experiência do usuário é um fator a ser levado em consideração. São diferentes aspectos positivos e negativos que um usuário pode ter ao se interagirem com um sistema computacional: satisfação, diversão, frustação, desafio, cansaço, entre outros. Não há como saber a experiência de uso de todo usuário.

3.2. Acessibilidade

Durante a interação, o usuário usa sua habilidade motora para agir sobre os dispositivos de entrada, seus sentidos (visão, audição e tato) e capacidade de percepção para identificar as repostas do sistema e sua capacidade de interpretação e raciocínio para compreender as respostas do sistema. Se a interface tiver algum tipo de barreira, isso pode impossibilitar o usuário a se interagir com o sistema.
A acessibilidade está relacionada à remoção de barreiras que possam dificultar mais usuários de interagirem com um determinado sistema. A intenção é incluir e não excluir. Cuidar da acessibilidade significa permitir que mais pessoas possam receber, compreender e utilizar o sistema. Isso significa que o sistema deve ser desenvolvido exclusivamente para uma determinada classe? Não. A seguir alguns cenários que demonstram a importância da acessibilidade.

3.2.1. Deficiência auditiva

Sr. X é deficiente auditivo que acessa a Internet sem grandes dificuldades. Um certo dia, seu provedor para de funcionar e o único meio de contato para tentar resolver o problema é por telefone. Todo o seu esforço para aprender o Português, a Língua Brasileira de Sinais (libras), não seria útil nesse caso.

3.2.2. Deficiência visual

Joana é uma jovem deficiente visual interessada em fazer o ENEM. Utilizando um leitor de telas, ela conseguiu acessar o site de inscrição do ENEM. Ela descobriu que precisava de identidade e CPF para realizar a inscrição, mas não conseguiu acessar um link para a inscrição e nem percebeu que as inscrições tinham terminado. A opção que abordava isso era uma figura que não pode ser lida pelo leitor de tela.

3.2.3. Aplicação genérica

A adequação de um sistema para um usuário leva em conta apenas pessoas com algum tipo de limitação? Imagine o GPS. Ele foi adaptado para motoristas, visto que o próprio (quando está dirigindo) não pode ficar toda hora

visualizando o sistema. Assim, o mecanismo de fala auxilia no processo de interação.
O governo brasileiro disponibiliza vários serviços federais por diversos meios e que devem ser acessados por todos, independentemente de alguma limitação. Desse modo, o governo fez o decreto presidencial de 2 de dezembro de 2004, que regulamenta as leis 10.048 e 10.098. Nele, o governo torna obrigatória a acessibilidade em sites do governo. Pode-se destacar algumas partes “Art. 47. No prazo de até doze meses a contar da data de publicação deste decreto, será obrigatória a acessibilidade nos portais e sítios eletrônicos da administração pública.

3.2.4. Comunicabilidade

A comunicabilidade de um sistema é a sua propriedade de transmitir ao usuário de forma eficaz e eficiente as intenções e princípios de interação que guiaram o seu design. A comunicabilidade está relacionada com o designer mostrar suas intenções de design e a lógica por trás da interface para o usuário. Um sistema interativo é resultado de um processo de design no qual um designer estabelece uma visão interpretativa sobre os usuários, seus objetivos, o domínio e o contexto de uso e toma decisões sobre como apoiá-los. O design deve comunicar ao usuário suas concepções e intenções sobre o que deseja que o usuário faça com o sistema, ou seja, como vai se comunicar com o sistema.
Faz parte da experiência diária das pessoas entender através de um ato de comunicação muitas mensagens subjacentes, como por exemplo perceber a filosofia de um autor através da sua peça de teatro. Da mesma forma, o objetivo da comunicabilidade é permitir que o usuário, através da sua interação com a aplicação, seja capaz de compreender as premissas, intenções e decisões tomadas pelo projetista durante o processo de design. Quanto maior o conhecimento do usuário da lógica do designer embutida na aplicação, maiores suas chances de conseguir fazer um uso criativo, eficiente e produtivo da aplicação. Junto com a usabilidade, a comunicabilidade pretende aumentar a aplicabilidade de software.
A comunicação envolve os processos que o designer cria que podem auxiliar os usuários a alcançarem seus objetivos (um passo a passo, por exemplo). Ela diz respeito à capacidade de a interface comunicar ao usuário a lógica do design: as intenções do designer e os princípios de interação resultantes das decisões tomadas durante o processo de design. Quanto melhor o usuário entender a lógica, melhor irá utilizar o sistema.
A lógica do design comunicada ao usuário deve refletir as decisões tomadas sobre: a quem se destina, para que serve, qual a vantagem de utilizá-la, como funciona. A analogia pode ser um recurso de comunicação utilizado para facilitar e aumentar a comunicabilidade. Permite comparar o uso de um software com base em experiências anteriores com softwares similares. Por exemplo, se usarmos dois medias players diferentes, devemos conseguir usar sem problemas os dois.

4. Interação Humano Computador (IHC)

Existem diversos atores envolvidos no desenvolvimento e uso dos sistemas computacionais interativos: fabricantes de hardware, de software, vendedores, profissionais de suporte manutenção, provedores de aceso à Internet, produtores de conteúdo, usuários, organizações, dentre outros. Com o aprofundamento do estudo dos aspectos que colocam o ser humano em uma posição mais "importante" em relação às interfaces do seu dia a dia, abre-se um campo muito importante de pesquisa e o início de uma nova realidade na era da computação. Do ponto de vista das aplicações Web, a interface de usuário deve ser entendida como uma parte do sistema computacional com a qual uma pessoa entra em contato físico, perceptiva e conceitualmente. Na Figura 4, temos uma representação gráfica de um modelo básico de IHC.

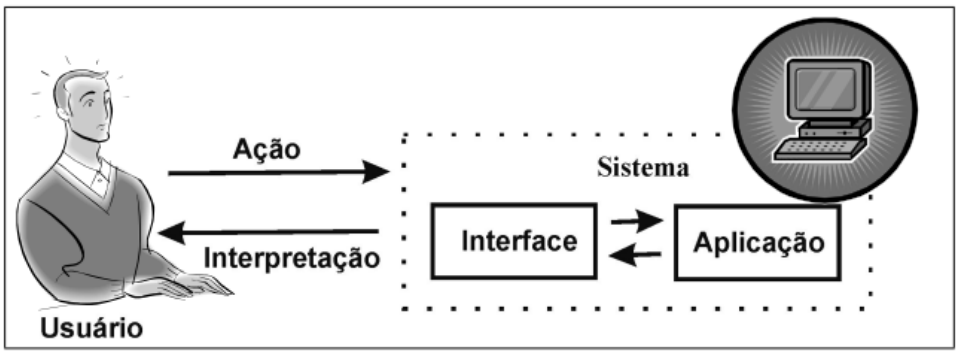

Figura 4 – Modelo básico IHC

A Interação Humano Computador (IHC) é uma subárea da Ciência da Computação preocupada com o projeto, avaliação e implementação de sistemas computacionais interativos para uso humano, bem como com o estudo dos principais fenômenos que envolvem estas etapas. Tem por objetivo principal fornecer aos pesquisadores e desenvolvedores de sistemas explicações e previsões para fenômenos de interação usuário-sistema e resultados práticos para o design da interface de usuário. Envolve o processo pelo qual os seres humanos interagem com os computadores, bem como os modos, meios ou processos envolvidos no uso de Tecnologias de Informação e Comunicação (TIC's). Os TIC's são sistemas interativos compostos por hardware, software e pelos diferentes meios de comunicação.

O objetivo é produzir sistemas utilizáveis e seguros, bem como sistemas funcionais. Nesse contexto, a posição do homem deixa de ser secundária e se torna o foco principal da interação e a interface passa a ser representativa a um modelo de mais fácil acesso pelo ser humano. A ideia desse novo modelo é criar interfaces com a maior adaptabilidade possível através de metas que podem ser resumidas em desenvolver ou melhorar segurança, utilidade, efetividade, eficiência, e usabilidade de sistemas no meio computacional.

Por exemplo, um usuário está mais interessado no aceso à Internet do que no dispositivo que possibilita o acesso ou nas peças que compõem o dispositivo ou como elas foram montadas. Apesar de as duas perspectivas serem sobre o mesmo dispositivo, a segunda perspectiva é mais comum a um desenvolvedor de hardware a um profissional de suporte.

Em outro exemplo, pensem uma organização que utiliza software como instrumento de trabalho. Para apoiar os processos de trabalho da organização, um gerente encomenda um sistema a uma empresa de desenvolvimento de software. Os desenvolvedores costumam se concentrar nas funcionalidades do software em como ele é estruturado internamente. Já os funcionários da organização geralmente se preocupam em como vão aprender e utilizar o software para realizar o seu trabalho com eficiência. Existe uma diferença sutil, porém importante, entre o que um sistema interativo deve permitir fazer (visão do cliente, responsável pela aquisição do sistema), o que ele de fato permite fazer (visão de quem produz, focada nas funcionalidades do software) e a maneira como ele é utilizado (visão dos usuários focada no impacto do software no seu trabalho ou na sua vida). A identificação dos diversos atores envolvidos e a articulação dos seus interesses e pontos de vista são importantes desafios no desenvolvimento de tecnologia.

4.1. Elementos Básicos de IHC

No contexto de IHC devemos considerar quatro elementos básicos: o sistema, os usuários, os desenvolvedores e o ambiente de uso (domínio de aplicação). Estes elementos estão envolvidos em dois processos importantes: a interação usuário-sistema e o desenvolvimento do sistema. O estudo de IHC identifica cinco enfoques para o estudo destes elementos e para a sua aplicação na melhoria dos processos de desenvolvimento e de interação usuário-sistema. Para cada um destes focos, diferentes disciplinas proporcionam os estudos teóricos que podem ser aplicados ao desenvolvimento. São eles:

- **Design e desenvolvimento do hardware e software**: estudo de tecnologias de dispositivos de entrada e saída; e tecnologias de software, como ambientes gráficos e virtuais.
- **Estudo da capacidade e limitação física e cognitiva dos usuários:** considera estudos de ergonomia para avaliar limites de esforço físico do usuário, e estudos de psicologia e ciência cognitiva sobre a capacidade humana de memorização, raciocínio e aprendizado.
- **Instrumentação teórica e prática para o design e desenvolvimento de sistemas interativos:** envolve o conhecimento teórico a respeito dos fenômenos envolvidos; modelos para o processo de desenvolvimento que descrevam as etapas necessárias e como devem ser conduzidas; diretrizes, técnicas, linguagens, formalismos e ferramentas de apoio a estas etapas.
- **Modelos de interfaces e do processo de interação usuário–sistema:** para desenvolver modelos abstratos do processo de interação compatíveis com as capacidades e limitações físicas e cognitivas dos usuários.
- **Análise do domínio e de aspectos sociais e organizacionais:** para avaliar o impacto que o contexto onde está inserido o usuário exerce sobre seus conhecimentos, sua linguagem e suas necessidades.

4.2. Interface e Interação

O termo interface é aplicado normalmente àquilo que interliga dois sistemas. Tradicionalmente, considera-se que uma interface homem-máquina é a parte de um artefato que permite a um usuário controlar e avaliar o funcionamento deste artefato através de dispositivos sensíveis às suas ações e capazes de estimular sua percepção. No processo de interação usuário-sistema a interface é o combinado de software e hardware necessário para viabilizar e facilitar os processos de comunicação entre o usuário e a aplicação. A interface entre usuários e sistemas computacionais diferencia-se das interfaces de máquinas convencionais por exigir dos usuários um maior esforço cognitivo em atividades de interpretação e expressão das informações que o sistema processa.
A interface é tanto um meio para a interação usuário-sistema, quanto uma ferramenta que oferece os instrumentos para este processo comunicativo. Nesse contexto, a interface de usuário pode ser entendida como a parte do sistema computacional com a qual uma pessoa entra em contato física, perceptiva e conceitualmente. Neste caso, a interface de usuário apresenta um componente físico, que o usuário percebe e manipula, e outro conceitual, que o usuário interpreta, processa e raciocina. Desta forma a interface é um sistema de comunicação.
A interface possui componentes de software e hardware (Fig. 5). Os componentes de hardware compreendem os dispositivos com os quais os usuários realizam as atividades motoras e perceptivas. Entre eles estão a tela, o teclado, o mouse e vários outros. O software da interface determina os efeitos no comportamento do sistema decorrentes das ações do usuário sobre o próprio sistema. Ele é a parte do sistema que implementa os processos computacionais necessários para:

- O controle dos dispositivos de hardware;
- Construção dos dispositivos virtuais (os widgets) com os quais o usuário também pode interagir;
- Geração dos diversos símbolos e mensagens que representam as informações do sistema;
- Interpretação dos comandos dos usuários.

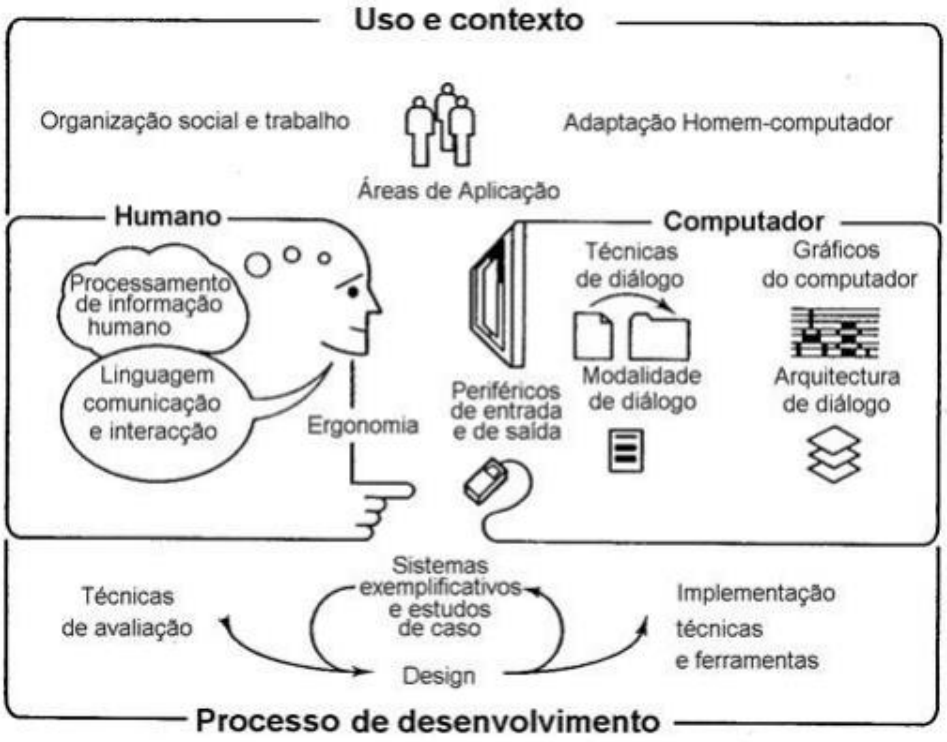

Figura 5 - Desenvolvimento da Interface

A interação representa a comunicação entre usuários e computadores ou outros tipos de produto. É um processo que engloba as ações do usuário sobre a interface de um sistema, e suas interpretações sobre as respostas reveladas por esta interface.

4.3. Affordance

Outra característica de uma interface é a revelação das affordances do sistema. As características físicas de um objeto evidenciam o que é possível fazer com ele e as maneiras que irá utilizá-lo. Affordance representa o conjunto de características de um objeto que indicam aos seus usuários as operações e manipulações que eles podem fazer com o objeto. As affordances fornecem fortes pistas ou indicações quanto à operação de objetos e quando se tira proveito delas, o usuário sabe exatamente o que fazer só olhando para o objeto.

A maneira como as pessoas percebem os objetos e já “sabem” o que fazem é importante para uma boa usabilidade. Por exemplo, ao fazer um cadastro, esperamos que tenham campos para serem preenchidos e um botão ao final com a palavra OK. Também existem as chamadas “Falsas Affordances”. Por exemplo, o usuário clica em um objeto pela sua percepção adquirida imaginando que efetuará determinada ação. No entanto, o objeto faz outra ação ou não faz ação nenhuma. Diante de um texto na Web o usuário encontra uma palavra sublinhada. Por conhecimentos adquiridos, tende-se a clicar em no texto imaginando que se trata de um link de uma página. No entanto, a palavra está apenas sublinha sem nenhum contexto. Isso é um exemplo de falsa affordance.

4.4. Perspectivas em IHC

Para compreender melhor as teorias de design de interface, precisamos entender as diferentes perspectivas que os sistemas de computador vêm atravessando ao longo do tempo (Fig. 6).
Inicialmente, o usuário era considerado uma máquina, que tinha que aprender a falar a linguagem do computador. Em seguida, com o surgimento da Inteligência Artificial, tentamos considerar o computador como uma pessoa. Nessas duas perspectivas, era fundamental dar poder ao sistema. Mais tarde, surgiu a perspectiva de computador como ferramenta, que o usuário utiliza para obter um resultado ou produto. Atualmente vemos outra mudança de perspectiva, na qual o computador é um mediador da comunicação entre pessoas. Nestas duas últimas perspectivas, o foco é no usuário, e não mais no sistema.

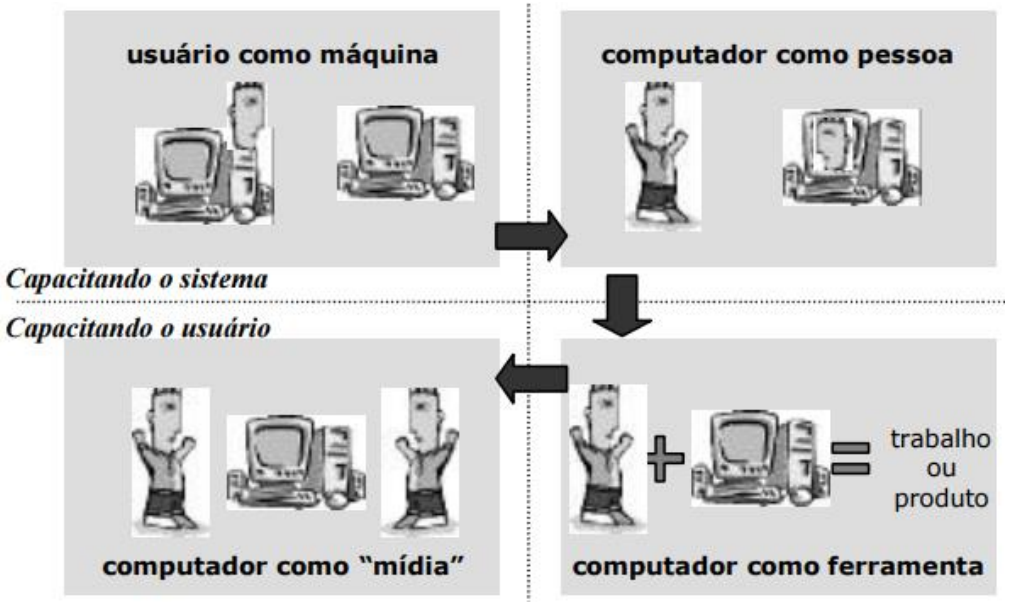

Figura 6 - Perspectivas em IHC

A interação e a interface devem ser adequadas para que os usuários possam aproveitar ao máximo o apoio computacional do sistema. Para isso, existem alguns critérios de qualidade: usabilidade, experiência do usuário, acessibilidade e comunicabilidade.

5. Testes em Webapp

O principal objetivo dos testes nas aplicações Web é observar a aplicação usando combinações de inputs e estados de forma a descobrir falhas no software. Estes testes necessitam ser vistos como um processo de desenvolvimento que é efetuado ao longo do projeto e não apenas no final, na entrega ao usuário.
A atividade de teste é uma tarefa complexa e por isso é normalmente dividida em fases com objetivos distintos, também conhecidas como níveis ou etapas de teste. Os principais níveis de teste encontrados na literatura são:

5.1. Teste de Unidade

Este teste tem como foco as menores unidades de uma aplicação que podem ser funções, procedimentos, métodos ou classe. No contexto das aplicações Web, a unidade pode ser considerada como uma página Web e seus elementos (campos, texto, formulários, botões, links, scripts) ou um serviço Web. Nesta etapa, cada unidade é testada separadamente e a atividade de teste pode ser aplicada à medida que ocorre a implementação sem a necessidade de dispor-se do sistema totalmente finalizado, sendo necessário em alguns casos a utilização de “drivers” ou “stubs” em substituição às funcionalidades ausentes. Por exemplo, um script de servidor é chamado por um “driver” que simula o navegador (“browser”) e recebe a página HTML como a saída resultante. Outro exemplo é um programa Java Script que valida os campos em uma página do lado do cliente e envia uma solicitação para um servidor fictício, simulados por um “stub”.

5.2. Teste de Integração

Normalmente realizado após as unidades terem sido testadas individualmente. À medida que as diversas partes são colocadas para trabalhar juntas, é preciso verificar se a interação entre elas funciona adequadamente. No contexto das aplicações Web, pode ser considerada a integração de páginas Web ou a composição de serviços Web.

5.3. Teste de Sistema

Em geral este teste é realizado quando o sistema está completo, com todas as suas partes integradas. O objetivo é verificar se os requisitos funcionais e não funcionais especificados foram corretamente implementados. O sistema é validado como um todo em um ambiente o mais semelhante possível do ambiente alvo real.

5.4. Teste de Aceitação

Este teste é geralmente realizado pelos usuários finais do sistema para avaliar se o produto entregue está de acordo com o solicitado. Em geral, o cliente instala e executa o aplicativo em seu próprio ambiente.

6. Técnicas de Teste e Critério de Cobertura

As principais técnicas de teste são: Funcional e Estrutural. O que distingue essencialmente as técnicas de teste é a fonte utilizada para definir os requisitos de teste e os critérios de cobertura utilizados para explorar, exercitar e avaliar as aplicações sob análise.

6.1. Teste Funcional

É uma técnica utilizada considerando o programa ou aplicação a ser avaliada como uma caixa preta. Para testá-lo, são fornecidas entradas e avaliadas as saídas geradas para verificar se estão em conformidade com os objetivos

especificados. Nessa técnica, os detalhes de implementação não são considerados. Os critérios de cobertura mais conhecidos da técnica de teste funcional são o Particionamento por Equivalência, Análise de Valor Limite, e Grafo Causa-Efeito.
Uma das vantagens dos critérios da técnica funcional é que eles requerem somente a especificação do produto para derivar os requisitos de teste, não sendo necessário o acesso ao código-fonte. Por outro lado, não é possível assegurar, por exemplo, que trechos críticos do código tenham sido percorridos e validados.
Em geral, a técnica de teste caixa preta para aplicações Web não é diferente da técnica utilizada em aplicações de software convencionais. Em ambos os casos o uso de um critério de cobertura e um adequado conjunto de requisitos de teste são definidos com base na especificação da funcionalidade dos itens que serão testados. Entretanto, algumas características específicas das aplicações Web podem afetar o projeto e a execução dos testes. Por exemplo, o teste de componentes gerados dinamicamente durante a execução pode ser uma tarefa mais árdua, devido à dificuldade de identificar e reproduzir as mesmas condições que cada componente produziu. Nesse caso, as especificações utilizadas para representar o comportamento de aplicações tradicionais deverão ser adaptadas para as aplicações Web.

6.2. Teste Estrutural

A técnica de teste estrutural, ou caixa branca, avalia o comportamento interno do componente de software. Essa técnica trabalha diretamente sobre o código fonte para avaliar aspectos tais como caminhos lógicos e conjuntos específicos de condições. Os critérios de cobertura associados à técnica estrutural são classificados com base na complexidade, no fluxo de controle e no fluxo de dados.
Dentre as limitações desta técnica, destaca-se o fato de que os requisitos de teste são baseados na estrutura interna da aplicação e desta forma eventuais problemas no código implicam em testes incompletos. No contexto das aplicações Web, critérios de testes do tipo caixa branca podem ser propostos:

- **Teste de todas as páginas** (all-pages) - onde cada página do aplicativo Web deve ser visitada nas atividades de teste pelo menos uma vez;
- **Teste de todos os Hyperlinks** (all-links) - onde cada link de cada página deve ser percorrido nas atividades de teste pelo menos uma vez;
- **Teste de todos os caminhos** (all-pathes) - onde todos os caminhos da aplicação Web deverão ser percorridos em algum teste pelo menos uma vez;
- **Teste de todos os usos** (all-use) - todos os caminhos de navegação serão exercidos considerando todas possibilidade de uso para todas definições de variáveis, formando uma dependência de dados.

Além das técnicas de caixa branca e caixa preta, temos ainda a técnica de teste caixa cinza. Trata-se de uma combinação entre as técnicas funcionais e estruturais, que considera tanto o comportamento da aplicação quando a sua estrutura interna. É esperado que a utilização desta técnica ajude a revelar problemas que não são facilmente capturados com uso individual das técnicas

funcionais e estruturais, especialmente problemas associados ao fluxo de informações e compatibilidades quanto à distribuição de hardware e configurações de software dos sistemas.

7. Processo de Testes

O processo de teste para Webapp engloba praticamente todas as etapas do processo de desenvolvimento de uma Webapp. Na Figura 7, temos uma pirâmide de projeto Webapp. Primeiramente é testado e revisto o conteúdo e o fluxo de teste ocorre da esquerda para a direita e de cima para baixo. O que é visível para o usuário fica no topo da pirâmide, seguindo pelos processos da estrutura, questões de instalação e implantação da Webapp.

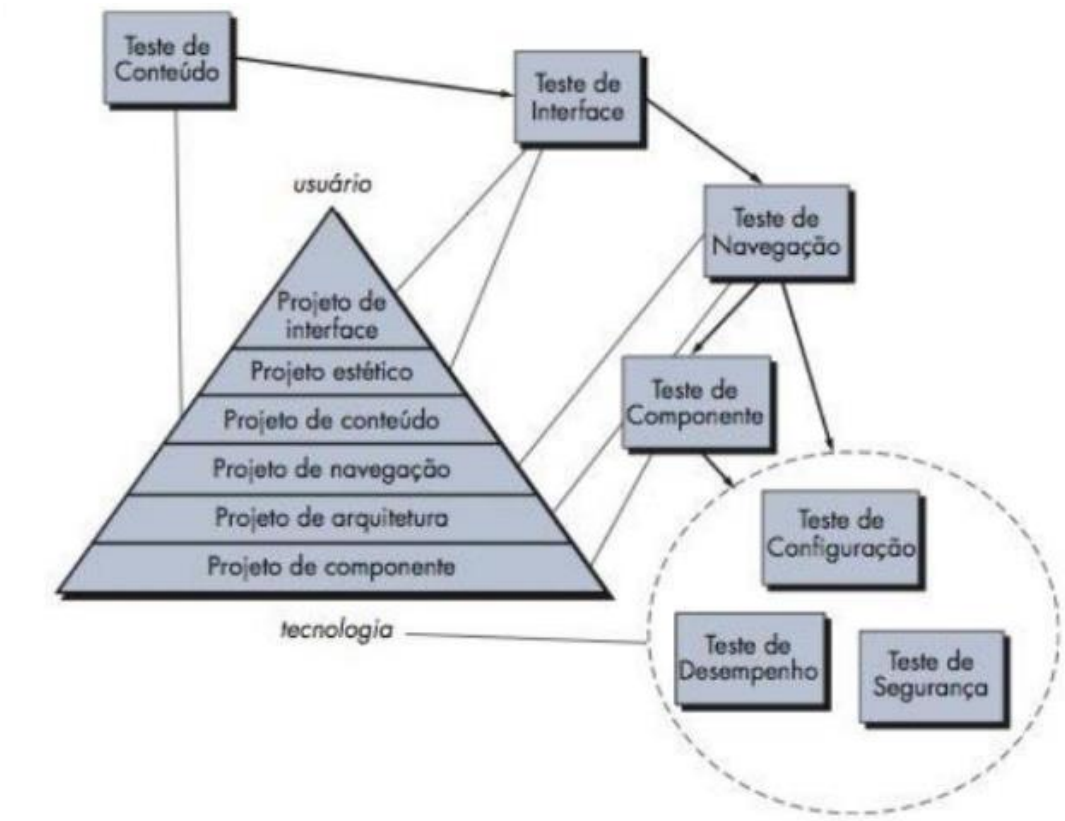

Figura 7 - Pirâmide de projeto Webapp - Fonte Pressman (2011)

- **Teste de conteúdo** - é revisto todo o conteúdo com a revisão de documentos e busca de erros, que podem ser desde erros tipográficos até informações incorretas, violação de lei e imagens sem direitos autorais. O teste de conteúdo busca descobrir erros de sintaxe (erros ortográficos e gramaticais) em um documento de texto, representações gráficas e outros meios, descobrir erros na exatidão ou integridade das informações em qualquer objeto que é visto na navegação e encontrar erros na estrutura ou organização do conteúdo fornecido para o usuário final;
- **Teste de interface** – começam com a análise e revisão da interface para garantir que esteja em conformidade com os requisitos dos interessados, incluindo análises do caso de uso. Durante o projeto são

estabelecidos critérios genéricos de qualidade para todas as interfaces de usuário. O próximo passo são os testes após o término do projeto, que experimentam mecanismos de interação e validam os aspectos estéticos da interface do usuário. O objetivo é encontrar erros relacionados com mecanismos da interface, na maneira como são implementadas as semânticas de navegação, funcionalidades da aplicação Web e exibição de conteúdo;

- **Teste de Navegação** - o objetivo é garantir que todos os mecanismos que um usuário pode navegar estejam em completo funcionamento e confirmar que cada semântica de navegação possa ser alcançada pela categoria de usuário;
- **Teste de Configuração** - visa testar um conjunto provável de configurações do cliente e do servidor para assegurar que a navegação seja a melhor possível e isolar possíveis erros. Segundo Pressman (2011), no lado do servidor é verificado se podem suportar as Webapp sem erro, incluindo servidores de banco de dados, SO, firewall etc. Alguns pontos importantes devem ser verificados: Se Webapp e SO são compatíveis; os arquivos, diretórios e dados de sistemas são criados quando a Webapp está operacional; firewall ou criptografia do servidor interfere no funcionamento ou velocidade da Webapp; a Webapp foi testada com configuração distribuída (se for o caso); a Webapp está adequadamente integrada com o software de banco de dados e sensível às suas atualizações; os scripts estão sendo executados corretamente; erros de administradores podem afetar a Webapp e até que ponto isto pode ocorrer; no lado cliente, o teste de configuração focaliza na compatibilidade da Webapp com configurações que contenham diferenças nos componentes de hardware (CPU, memórias armazenamento e dispositivos de impressão), Sistema Operacional, navegadores e componentes da interface de usuário (Active X, Java applets e outros), plug-ins (Quick Time, Real Player, Flash Player) e outros e elementos de conectividade de rede (TIC);
- **Teste de Segurança** - focaliza três elementos distintos que são o servidor, que disponibiliza a Webapp, o caminho de comunicação entre o servidor e o cliente, e o ambiente do lado do cliente. O teste focaliza o acesso não autorizado ao conteúdo e funcionalidades da Webapp que juntamente com o servidor e o ambiente representam um alvo para hackers e outras pessoas que busquem ter acesso indevido a Webapp, roubando informações, modificando conteúdos, degradando o desempenho, desativando funcionalidades etc.;
- **Teste de Desempenho** - visa descobrir problemas que possam afetar o acesso a requisição do usuário e que podem estar relacionados a falta de mecanismos do lado do servidor, largura de banda inadequada, banco de dados inadequados, sistemas operacionais, funcionalidades da Webapp mal projetadas e problemas de hardware ou software.