Unidade II

3 ENGENHARIA DE *SOFTWARE* E INTERAÇÃO HUMANO–COMPUTADOR

A Engenharia de *Software* tem o foco voltado para o produto e seu processo de desenvolvimento. A Interação Humano-Computador tem o foco voltado à interação do ser humano e da máquina. As duas áreas, no entanto, estabelecem métodos e processos de desenvolvimento de sistemas interativos.

3.1 Visão da Engenharia de *Software* e visão da Interação Humano–Computador

Existe uma diferença fundamental entre as abordagens adotadas pelos engenheiros de *software* e pelos especialistas em Interação Humano-Computador. Segundo Brown (1996), enquanto os engenheiros de *software* têm o foco voltado para o produto e seu processo (centrado em sistema), os especialistas em Interação Humano-Computador têm o foco mais direcionado aos aspectos de interação do ser humano e da máquina (centrado no usuário). Em sua pesquisa, Brown (1996) relata que as metodologias de Engenharia de *Software* são úteis para especificar e construir os aspectos funcionais de um sistema de *software*. No entanto, especialistas em Interação Humano-Computador mostram uma compreensão melhor do usuário, priorizando um entendimento aprofundado das características deste usuário e uma consciência das tarefas que um usuário tem de executar. Especialistas de Interação Humano-Computador testam ideias de *design* em usuários reais e usam técnicas de avaliação formais, substituindo o *design* da interface guiado pela intuição.

Lembrete

A Engenharia de *Software* surgiu com o objetivo de melhorar o processo de desenvolvimento de *software*, bem como a qualidade do produto de *software*.

Segundo Rozanski e Haake (2003), Interação Humano-Computador (IHC) tornou-se um componente essencial para todos os profissionais de computação. Cientistas da computação e engenheiros de *software* também precisam entender os princípios e conceitos de IHC; ainda que não sejam os principais profissionais responsáveis pela compreensão do usuário e pelo projeto da interface, trabalharão com os profissionais responsáveis.

Lembrete

Interação Humano-Computador (IHC) é uma disciplina que se preocupa com o *design*, a avaliação e a implementação de sistemas computacionais

> interativos para uso humano e com o estudo dos principais fenômenos que os cercam (ACM SIGCHI, 1992).

Contudo, as duas áreas propõem o desenvolvimento de sistemas interativos de forma sistemática, definindo modelos de ciclo de vida, métodos e técnicas (SILVA *et al.*, 2004).

Paula, Barbosa e Lucena (2005) alertam para a importância da comunicação entre as áreas de IHC e Engenharia de *Software*. Os projetistas de IHC precisam levar suas preocupações e decisões de forma clara aos engenheiros de *software* e vice-versa, para que, juntos, cheguem à solução final. Ainda segundo os autores, ambas as áreas tratam da qualidade do produto final, no entanto sob perspectivas e focos diferentes. A IHC focaliza interação e *design* de interface do usuário, levando em conta as necessidades, os valores e as expectativas dos usuários, visando à qualidade de uso da solução projetada. A Engenharia de *Software* tem foco no projeto e na especificação da funcionalidade interna do sistema, bem como em sua arquitetura, visando à qualidade estrutural do produto de *software* final.

O quadro a seguir apresenta um resumo das práticas tradicionais da Engenharia de *Software* e as melhores práticas no desenvolvimento centrado no usuário da IHC, para o desenvolvimento de sistemas interativos.

Quadro 1 – Práticas em Engenharia de *Software* e projeto centrado no usuário

Práticas tradicionais no desenvolvimento de *software*	Melhores práticas no desenvolvimento centrado no usuário
Desenvolvimento dirigido à tecnologia.	Dirigido ao usuário.
Foco em componentes de sistema.	Foco na solução para o usuário.
Contribuição individual.	Equipe multidisciplinar, incluindo usuários, clientes, especialistas em fatores humanos etc.
Foco nas características internas da arquitetura.	Foco em atributos externos (interação, aparência e funcionamento – *look and feel*)
Qualidade medida por fatores como defeitos de produto e desempenho (qualidade de sistema).	Qualidade definida por satisfação do usuário e desempenho (qualidade em uso).
Implementação antes de validação humana.	Implementação com base nas avaliações e aprovações dos usuários.
Soluções são produzidas a partir de requisitos funcionais (características e recursos do sistema)	Entendimento do contexto de uso (o usuário, a tarefa e o ambiente de trabalho).

Adaptado de: Seffah e Metzker (2004, p. 103).

Relevantes autores da área de Engenharia de *Software* mostraram a importância da interface de usuário. Sommerville (2003) relata que a interface de usuário é uma das características que vêm sendo priorizadas, com enfoque especial no usuário. Os autores Peters e Pedrycz (2001) afirmam que os fatores humanos, bem como a inclusão do usuário no desenvolvimento de interfaces, constituem uma fronteira relativamente nova da Engenharia de *Software*. O fato é que, atualmente, a interface de usuário vem ganhando ainda mais importância nos projetos de sistemas interativos. Entretanto, a inclusão do usuário no desenvolvimento de interfaces, embora já não seja uma fronteira tão nova assim na Engenharia de *Software*, ainda não acontece consoante sua importância.

Existem vários modelos de ciclo de vida para o desenvolvimento de sistemas interativos apresentados na literatura de IHC. Segundo Rocha e Baranauskas (2003), os modelos de ciclo de vida de *design* em IHC "envolvem desde uma discussão crítica dos ciclos de vida clássicos para o desenvolvimento de software, originais da Engenharia de Software, até modelos mais específicos do ciclo de vida de *design*, como, por exemplo, o modelo estrela", que será apresentado posteriormente.

No entanto, os modelos de ciclo de vida apresentados pela IHC não discutem as etapas para o desenvolvimento e a implementação do sistema e de suas funcionalidades, diferentemente dos modelos de ciclo de vida da Engenharia de *Software*. Estes modelos enfocam o desenvolvimento da interação com o usuário.

De acordo com Sharp, Rogers e Preece (2005), o termo *design* não deve ser confundido com o *design* comumente traduzido como **projeto** e utilizado pela comunidade de Engenharia de *Software* para uma fase do modelo de ciclo de vida de *software*. Do mesmo modo, Leite (1998), destaca que a atividade de *design* da interface do usuário não deve ser confundida com a atividade de especificação da maneira proposta na Engenharia de *Software*, nem com o processo de construção do *software*.

Segundo Souza *et al.* (1999), o *design* de interfaces de usuário é uma atividade que requer análise dos requisitos dos usuários e suas tarefas, concepção, especificação e prototipação da interface, e avaliação da utilização do protótipo pelos usuários.

Observação

Requisitos de *software* são características, propriedades e comportamentos desejáveis para um produto de *software*. Eles podem ser divididos em (SWEBOK, 2004):

- Requisitos funcionais: estão diretamente ligados à funcionalidade do *software*. Descrevem as funções que o *software* deve executar.
- Requisitos não funcionais: expressam as restrições a que o *software* deve atender ou as qualidades específicas que o software deve ter. Às vezes são conhecidos como restrições ou requisitos de qualidade.

No entanto, para Marcus (2002), o *design* de interface de usuário ainda não tem uma definição consolidada. Para ele, deveria ser chamado de **desenvolvimento de interface de usuário**, semelhante ao desenvolvimento de *software*, adotado pela Engenharia de *Software*.

Observação

O termo *design* é encontrado na literatura sobre IHC como algo mais amplo em relação ao mesmo termo utilizado na Engenharia de *Software*. Entretanto, neste livro-texto, o termo, algumas vezes, aparecerá traduzido para **projeto**, mesmo quando utilizado nas atividades relacionadas à IHC.

3.2 Modelos de ciclo de vida de *software* tradicionais da Engenharia de *Software*

Segundo Pressman (2006), nos últimos cinquenta anos, o *software* – programas, dados e documentos – evoluiu. Passou de um ferramental especializado em solução de problemas e análise de informações para um produto de indústria. No entanto, ainda há problemas em produzir *software* de alta qualidade e dentro do prazo e do orçamento estabelecido. Desse modo, o intuito da Engenharia de *Software* é fornecer uma estrutura para a construção de *software* com alta qualidade.

O Institute of Electrical and Electronic Engineers (IEEE) definiu Engenharia de *Software* como: "(1) A aplicação de uma abordagem sistemática, disciplinada e quantificável, para o desenvolvimento, operação e manutenção do *software*; isto é, a aplicação da engenharia ao *software*. (2) O estudo de abordagens como as de (1)" (IEEE, 1990).

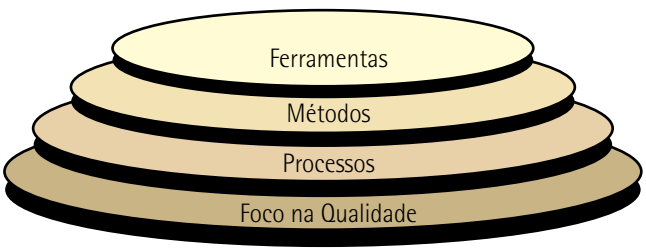

Figura 2 – Camadas da Engenharia de *Software*

Segundo Pressman (2006), a Engenharia de *Software* é uma tecnologia composta por camadas, conforme apresentado na figura. É uma disciplina que tem como base de apoio o **foco na qualidade**. A camada de **processos** assegura a junção das camadas de tecnologia e proporciona o desenvolvimento racional e oportuno de *software* de computador. Os **métodos** de Engenharia de *Software* fornecem a técnica de como fazer para construir *software*. As **ferramentas** de Engenharia de *Software* fornecem apoio automatizado ou semiautomatizado para os processos e para os métodos.

De acordo com estas definições, a Engenharia de *Software* surgiu com o objetivo de melhorar o processo de desenvolvimento de *software*, bem como a qualidade do produto de *software*.

A Engenharia de *Software* propõe vários modelos de ciclo de vida de *software* (também conhecidos como paradigmas de processo ou **modelos de processo de *software***).

A partir de IEEE (1990), Paula Filho (2008), Pressman (2006), Schmidt (2000), Sommerville (2003) e Swebok (2004), podem-se apresentar as fases tipicamente utilizadas nos modelos de ciclo de vida da engenharia de *software*, aplicáveis à maioria dos projetos de *software*. A seguir, apresenta-se a descrição resumida de casa fase:

- Requisitos: esta fase, também chamada de Levantamento de Requisitos, visa obter um conjunto de requisitos de um produto acordado entre cliente e fornecedor. Sua finalidade é definir o que o sistema deve fazer. A identificação de questões relacionadas a algum atributo de qualidade, tais como desempenho, confiabilidade, disponibilidade e segurança, também faz parte desta fase.

- Análise: nesta fase o objetivo é detalhar, estruturar e validar os requisitos de *software* levantados durante a fase de Requisitos, de forma que estes possam ser usados como base para planejamento e acompanhamento detalhados da construção do produto. Enquanto o Levantamento de Requisitos focaliza a visão que os usuários têm dos requisitos do *software*, a Análise dos Requisitos focaliza a visão dos desenvolvedores, ainda considerando apenas o que fazer, sem entrar no espaço das soluções.

- Projeto (*Design*): sua finalidade principal é decidir como o sistema será implementado. Uma arquitetura de *software* é definida durante esta fase e estabelece a estrutura com que o produto de *software* deverá ser implementado para satisfazer os requisitos. A arquitetura inclui o projeto do banco de dados e o desenho interno que modela as partes lógicas e físicas do *software*, bem como suas interconexões e comunicações com *softwares* externos. Durante a fase de projeto, decisões táticas e estratégicas são tomadas para atender aos requisitos funcionais e de qualidade de um sistema.

- Implementação: na fase de implementação o projeto é transportado para uma linguagem de implementação em forma de código-fonte. O objetivo desta fase é traduzir a solução em código. A conclusão desta fase somente ocorre quando todo o código está escrito e documentado, compilado, livre de erros e seguindo o padrão do projeto. Além disso, até o final desta fase, um plano de testes (descrevendo quando e como testar cada parte do código) deve ser traçado.

- Testes: nesta fase o objetivo é integrar e testar o sistema. Durante a fase de testes, o sistema é verificado para certificar-se de que os requisitos especificados anteriormente foram todos implementados corretamente. O código gerado deve ser testado rigorosamente baseado nos requisitos analisados.

- Implantação: tem a finalidade de assegurar uma transição bem-sucedida do sistema desenvolvido para seus usuários. Consiste na instalação do produto de *software* no ambiente designado e a revisão e teste de aceitação por parte do cliente do produto de *software* (*installation of the software product in the target environment and the acquirer's acceptance review and testing of the software product*). Nesta fase estão incluídos artefatos como material de treinamento e procedimentos de instalação.

Essas fases podem se sobrepor ou podem ser executadas iterativamente (IEEE, 1990).

Os principais e mais tradicionais modelos de ciclo de vida de *software* são:

- cascata;

- incremental;

- espiral;

- prototipagem.

Observação

A Engenharia de *Software* apresenta vários modelos de ciclo de vida de *software*. Os modelos abordados por meio de conceitos básicos neste livro-texto são para fazer um paralelo com os modelos da IHC que serão apresentados nas próximas páginas.

Saiba mais

Para saber mais sobre os modelos de ciclo de vida de *software* apresentados pela Engenharia de *Software*, consulte também:

PRESSMAN, R. S. *Engenharia de software*. 7. ed. Porto Alegre: AMGH, 2011.

SOMMERVILLE, I. *Engenharia de software*. 8. ed. São Paulo: Pearson, 2007.

3.2.1 Modelo em Cascata

Um dos primeiros modelos propostos foi o Modelo Cascata, também chamado de Ciclo de Vida Clássico, apresentado na figura a seguir:

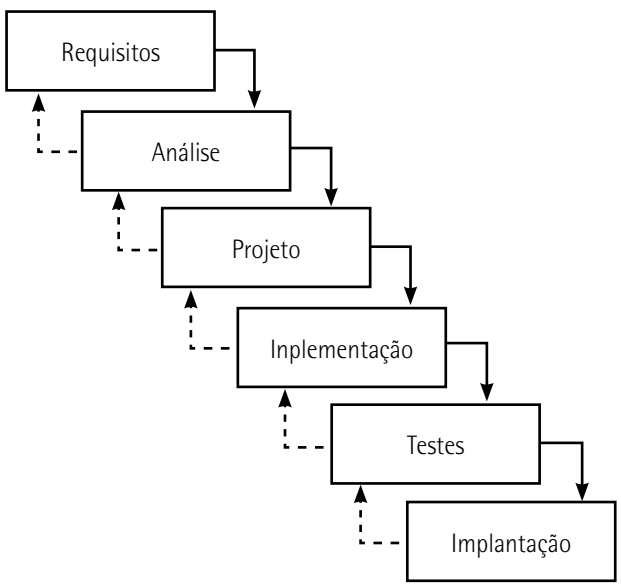

Figura 3 – O Modelo em Cascata

O Modelo em Cascata sugere uma abordagem sistemática e sequencial – fases (ou estágios) de desenvolvimento são apresentadas em sequência – para o desenvolvimento de *software*. O processo começa com a especificação dos requisitos pelo cliente e progride ao longo das outras

atividades fundamentais. O desenvolvimento de uma fase deve terminar antes de a próxima começar. Desse modo, só quando todos os requisitos forem enunciados pelo cliente, analisados e documentados, é que a equipe de desenvolvimento poderá realizar as atividades de projeto de sistema (PFLEEGER, 2004; PRESSMAN, 2006; SOMMERVILLE, 2003).

O problema do Modelo em Cascata é sua divisão rígida do projeto nesses estágios distintos. Estabelecer acordos e requisitos bem-definidos no estágio inicial do processo é uma tarefa difícil, pois os requisitos do cliente, por natureza, sempre se modificam. Outro problema é a baixa visibilidade para o cliente, que somente verá o resultado no final do projeto.

3.2.2 Modelo Incremental

O Modelo Incremental, conforme apresentado na figura a seguir, combina as vantagens do Modelo em Cascata com as vantagens do modelo evolucionário (PRESSMAN, 2006; SOMMERVILLE, 2003). Neste modelo, que tem o objetivo de apresentar um produto operacional a cada incremento, os requisitos e conceitos de *software* e sistema são, primeiro identificados em um esboço pelos clientes, e, em seguida, é definida uma série de estágios de entrega, com cada estágio fornecendo um subconjunto das funcionalidades do sistema. Esses estágios são repetidos cada vez que há uma nova versão do *software*.

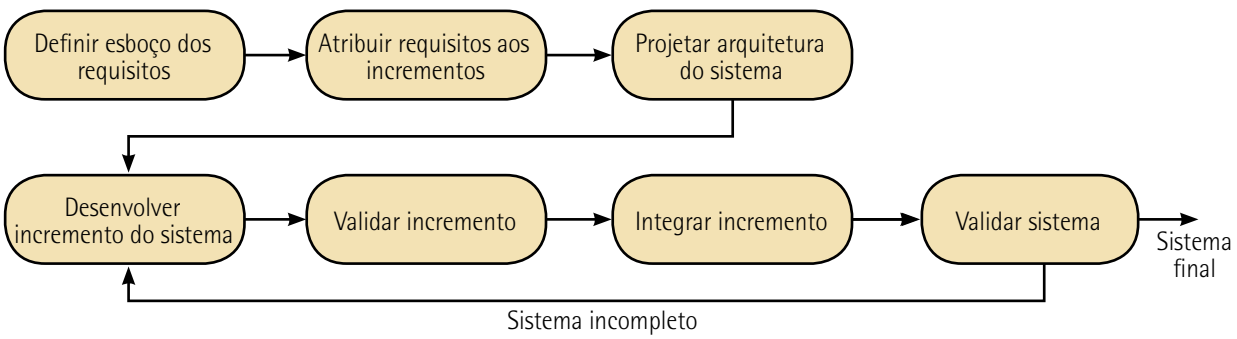

Figura 4 – Desenvolvimento incremental

Os primeiros incrementos já oferecem aos usuários condições de colocá-los em operação, bem como de experimentar o sistema, possibilitando o esclarecimento e a definição de requisitos para os próximos incrementos.

3.2.3 Modelo Espiral

O Modelo Espiral, apresentado na figura a seguir, originalmente proposto por Boehm (1988), é um modelo evolucionário de processo de *software* que combina a natureza iterativa da prototipagem com os aspectos controlados e sistemáticos do Modelo em Cascata (PRESSMAN, 2006). O processo é representado como uma espiral, e não como uma sequência de atividades, com algum retorno de uma atividade para outra (SOMMERVILLE, 2003). O produto é desenvolvido em uma série de iterações, de forma que cada nova iteração corresponde a uma volta na espiral. Desse modo, é possível construir produtos em prazos curtos, com novas características e recursos que são agregados à medida que a experiência descobre sua necessidade (PAULA FILHO, 2008).

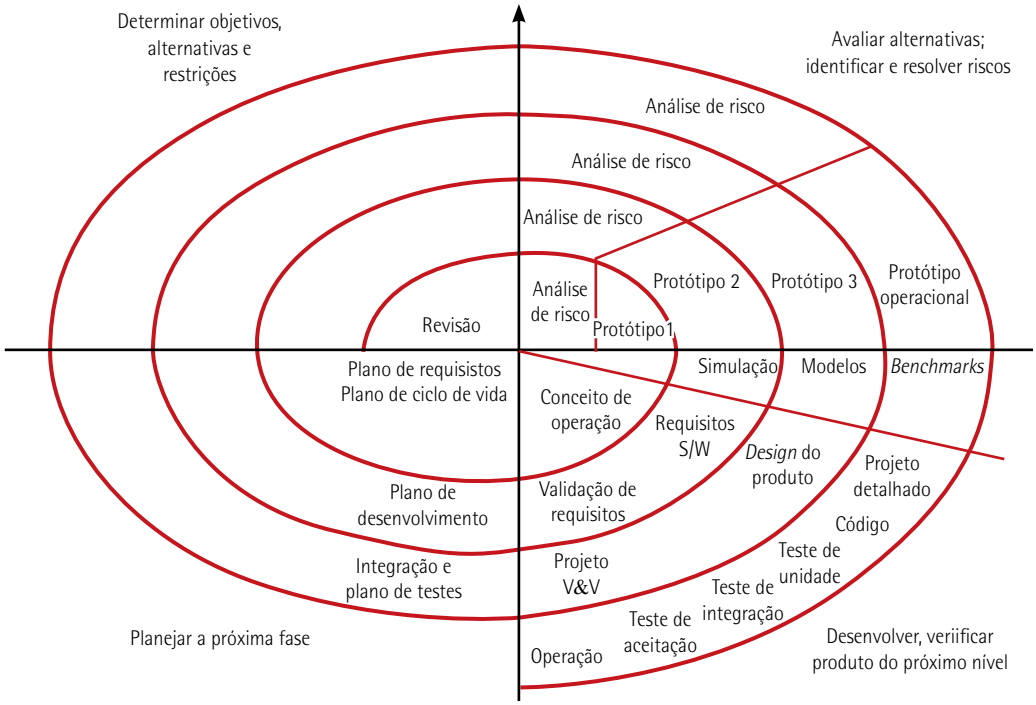

Figura 5 – Modelo Espiral

O modelo abrange de forma clara o gerenciamento de riscos e se aplica ao desenvolvimento de sistemas e *software* de grande porte (BOEHM, 1988).

3.2.4 Prototipagem

A prototipagem (ou prototipação) permite que todo o sistema, ou parte dele, seja construído rapidamente para que questões sejam entendidas ou esclarecidas.

De acordo com Sharp, Rogers e Preece (2005), os protótipos são muito úteis quando se está discutindo ideias com *stakeholders*, são dispositivos que facilitam a comunicação entre os membros das equipes e que consistem em uma maneira eficaz de testar as ideias para você mesmo.

Os protótipos de *software* são produzidos com funcionalidade e desempenho limitados (PETERS; PEDRYCZ, 2001), cobrindo cada vez mais requisitos, até que se atinja o produto desejado. Pfleeger (2004, p. 42) compara o modelo de prototipagem com o protótipo de engenharia. Segundo o autor, o objetivo é o mesmo, ou seja, é eficaz "quando os requisitos ou projeto necessitam de investigações repetidas para garantir que desenvolvedor, usuário e cliente cheguem a um consenso sobre o que é necessário e o que é proposto".

Uma vantagem desse modelo é solucionar o problema da espera no Modelo em Cascata, ou seja, não é necessário esperar até o final do ciclo de desenvolvimento para poder obter uma versão operacional do *software* (PETERS; PEDRYCZ, 2001). Outra vantagem desse modelo é deixar o usuário mais tranquilo ao "ver" o progresso do desenvolvimento do sistema.

Do ponto de vista da Engenharia de *Software*, a prototipagem é parte fundamental do processo de projeto de interface com o usuário (PFLEEGER, 2004; SOMMERVILLE, 2003).

O paradigma de prototipagem pode ser dividido em evolucionário e descartável (*throw-away*). Na prototipagem evolucionária, o planejamento de protótipos é mais cuidadoso, isto porque os protótipos não serão descartados. Neste modelo a equipe de desenvolvimento trabalha com os *stakeholders* os aspectos mais visíveis do sistema, normalmente a interface de usuário, sendo especialmente útil quando os usuários possuem dificuldade para expressar os requisitos do sistema.

Na prototipagem evolucionária, executam-se simultaneamente as fases de análise, projeto e implementação a fim de desenvolver, de forma rápida, uma versão simplificada do sistema proposto e apresentá-la aos *stakeholders* para avaliação e retorno de informações, conforme apresentado na figura a seguir (DENNIS; WIXOM; ROTH, 2014). Caso existam requisitos que ainda necessitem ser refinados, uma nova iteração será realizada. Neste caso, a equipe de projeto novamente analisa, projeta e volta a implementar outro protótipo, com novos recursos e correção das deficiências encontradas.

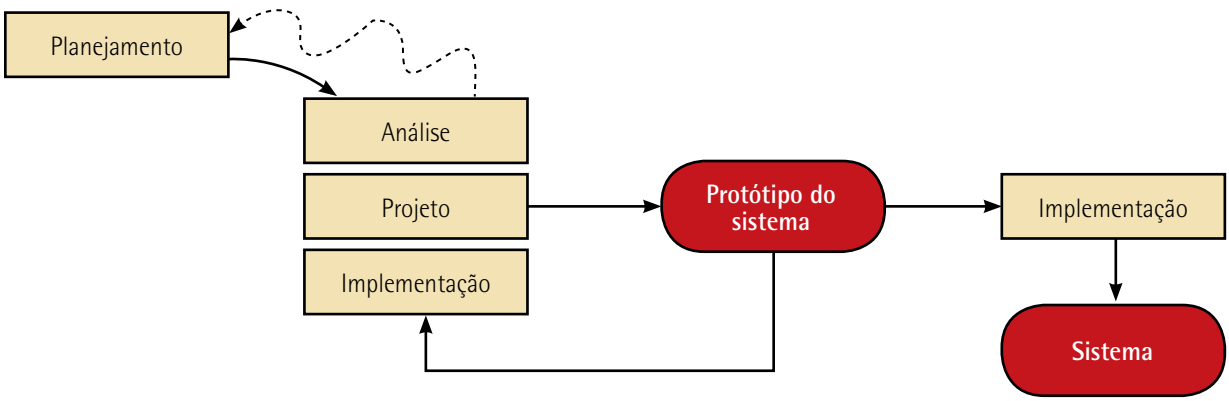

Figura 6 – O Modelo de Prototipagem

A prototipagem descartável (*throw-away*) consiste na construção de protótipos utilizados para explorar as alternativas de projeto, entender melhor os requisitos do sistema e reduzir riscos.

Conforme apresentado na figura a seguir, a prototipagem descartável tem uma fase de análise mais minuciosa, usada para analisar os requisitos e desenvolver ideias para a concepção do sistema. Entretanto, sugestões e solicitações de novos recursos pelos usuários e questões técnicas complexas podem não ter sido bem-compreendidos pela equipe de projeto. Para que se tenha, portanto, o correto entendimento, essas questões são examinadas pela análise, concepção e construção de um protótipo de *design* (DENNIS; WIXOM; ROTH, 2014).

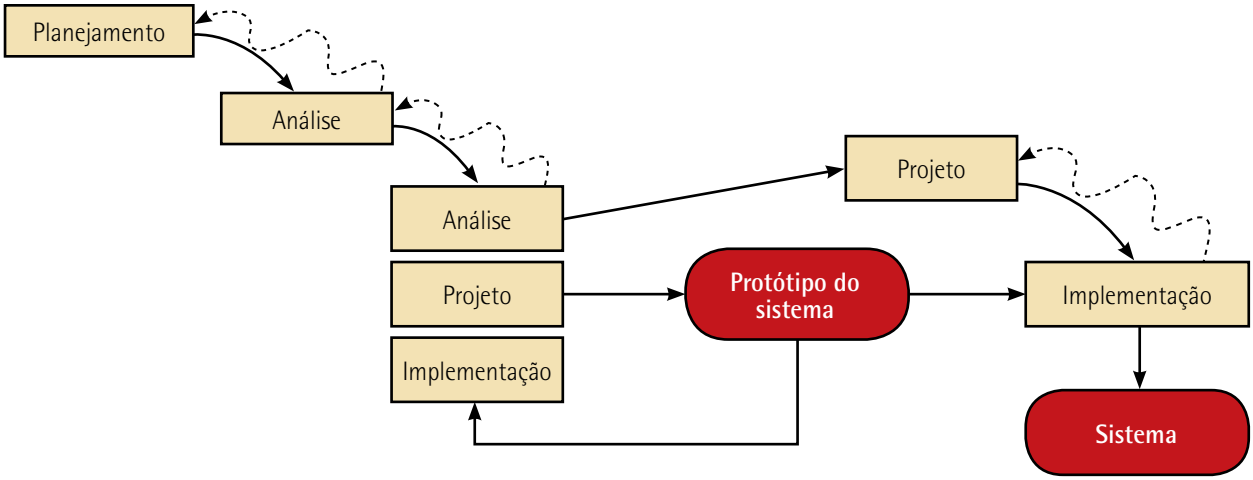

Figura 7 – Modelo de prototipagem descartável

O protótipo gerado nesse modelo não tem o objetivo de ser um sistema funcional. Depois de cumprir com seu propósito, ele será descartado.

3.3 Modelo de ciclo de vida de *design* de interface de usuário

3.3.1 Modelo Estrela

Proposto por Hix e Hartson em 1989, o Modelo Estrela, apresentado na figura a seguir, emergiu de um trabalho empírico realizado por ambos, observando como os projetistas de interface trabalhavam (SHARP; ROGERS; PREECE, 2005).

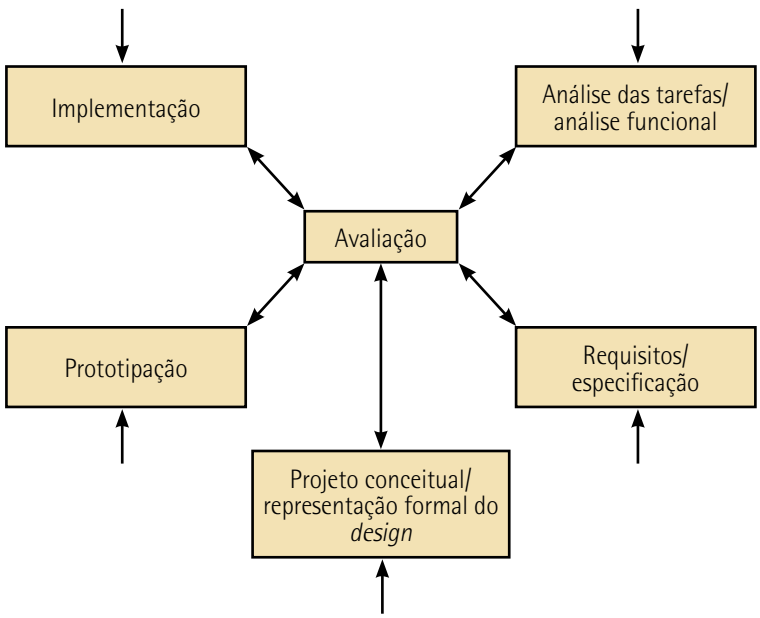

Figura 8 – Modelo Estrela

Segundo Costabile (2001), enquanto o modelo em Cascata sugere uma abordagem *top-down* (descendente, partindo da visão do sistema para a do usuário – mais formal), o modelo estrela reconhece que esta abordagem precisa ser complementada por uma abordagem *bottom-up* (ascendente, partindo da visão do usuário para a do sistema – mais criativa). Diferentemente dos modelos propostos pela Engenharia de *Software*, o modelo estrela não especifica ordenamento algum das atividades.

Um projeto pode começar de qualquer atividade (ponta na estrela) e seguir por qualquer outra, no entanto, deve sempre passar pela atividade de avaliação. Ou seja, o modelo possui uma avaliação central e, sempre que uma atividade for completada, seu trabalho deverá ser avaliado. Desse modo, os requisitos, o projeto e o produto evoluem gradualmente.

No entanto, esse modelo apresenta como desvantagem uma baixa visão gerencial para os gerentes e desenvolvedores. Uma explicação pode estar no fato do modelo estrela ser extremamente flexível (SHARP; ROGERS; PREECE, 2005).

3.3.2 Engenharia de Usabilidade

Engenharia de Usabilidade, segundo Rocha e Baranauskas (2003), é um termo usado para definir o processo de projeto de sistemas computacionais que visam à facilidade de aprendizado e de uso, e que sejam agradáveis para as pessoas. Além disso, propõe a aplicação de métodos empíricos ao projeto de sistemas baseados em computador.

Segundo Brown (1996), Engenharia de Usabilidade é um modelo com foco principal em conhecer o usuário e suas tarefas, bem como testar com os usuários os protótipos de um produto em desenvolvimento.

O usuário e o projetista de interface são os membros fundamentais nesta metodologia. No entanto, esta metodologia não apresenta uma relação definida entre o projetista de interface e o engenheiro de *software*.

Na figura a seguir são ilustradas as fases e o fluxo das atividades no modelo de Engenharia de Usabilidade proposto por Nielsen (1992, 1993) e sumarizado por Rocha e Baranauskas (2003). O modelo possui três fases: pré-projeto, projeto e pós-projeto.

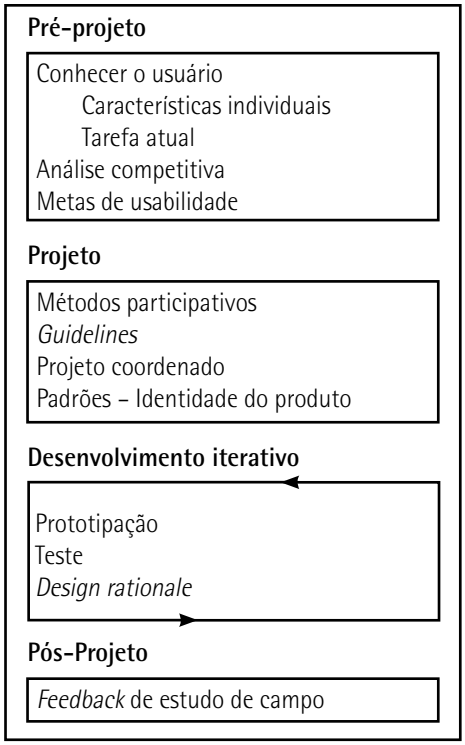

Figura 9 – Modelo de Engenharia de Usabilidade

Na fase de pré-projeto, o objetivo é a busca de informações para compreensão do usuário, das suas atividades e do seu contexto de trabalho, além das funcionalidades a serem implementadas no sistema. Também nessa fase são realizados estudos para uma análise comparativa de produtos existentes e testes com usuários no uso desses produtos. As metas de usabilidade são definidas ainda nesta fase.

A fase do projeto comporta o projeto inicial e o desenvolvimento iterativo. O projeto inicial é constituído da especificação inicial da interface. O *design* participativo é realizado, e o uso de *guidelines* é recomendado. O projeto coordenado (desenvolvimento paralelo da funcionalidade, da interface, do *help* e do material de treinamento) faz parte desta fase. O uso de padrões aumenta o reúso de código e facilita a documentação. O desenvolvimento iterativo é alimentado por *feedback* de testes até que os objetivos tenham sido alcançados. Os objetivos dessa fase são produzir um protótipo com princípios de usabilidade e verificar empiricamente, com usuários reais, se as metas foram atingidas. O *design rationale*, além de manter a memória do processo de projeto, ajuda a manter a consistência ao longo das diferentes versões do produto.

Por fim, a última fase, pós-projeto, tem como objetivo conduzir estudos de campo do produto em uso a fim de obter dados para próximas versões e produtos futuros.

Nielsen (1992) reconhece que frequentemente orçamentos ou restrições de tempo não permitirão o uso de todas as atividades do Modelo de Engenharia de Usabilidade. No entanto, recomenda um mínimo:

- visite os locais de trabalho do cliente antes do início do projeto;

- faça um projeto iterativo com métodos participativos;
- use prototipação e testes empíricos com usuários reais.

3.3.3 Projeto centrado no usuário

A IHC enfatiza a necessidade de uma abordagem centrada no usuário. Segundo Sharp, Rogers e Preece (2005), o fio condutor do desenvolvimento de um produto deveria ter como base os usuários reais e suas metas, não apenas a tecnologia. Desse modo, para se ter um sistema bem-projetado, os projetistas deveriam extrair o máximo da habilidade e dos julgamentos humanos mais importantes para o trabalho em questão. Portanto, deveriam apoiar o usuário, e não limitar suas ações.

Para Faulkner e Culwin (2000), as necessidades do usuário devem ser consideradas desde as fases iniciais do projeto, e o produto (interface do usuário) deve ser construído por meio da abordagem centrada no usuário.

Os princípios do projeto centrado no usuário, ou *User-Centered Design* (UCD), são focalizar desde o início os usuários e as tarefas que desenvolvem num determinado ambiente, medir a utilização do produto observando a interação do usuário com ele e utilizar um processo de *design* iterativo, no qual o *design* possa ser modificado após as fases de prototipação ou testes (SOUZA; COSTA; SPINOLA, 2006).

> Projeto centrado no usuário é uma atividade multidisciplinar que incorpora fatores humanos e conhecimento de ergonomia e técnicas com o objetivo de aumento da eficácia e eficiência, melhorando as condições humanas de trabalho, segurança, desempenho e evitar possíveis efeitos contra a saúde do homem (BEVAN, 1999 *apud* SOUZA; COSTA; SPINOLA, 2006, p. 5).

3.3.4 *Design* Participativo

O *Design* Participativo (DP) é uma abordagem que envolve ativamente o usuário. Segundo Sharp, Rogers e Preece (2005), a ideia dessa abordagem surgiu na Escandinávia, no final dos anos 1960 e início dos 1970. Sua intenção consiste em fazer que os usuários se tornem parceiros como os outros na equipe de projeto, participando de todas as atividades de desenvolvimento.

A inclusão do usuário final no processo de desenvolvimento se deve ao seu conhecimento das rotinas de trabalho, além de servir como fonte de informação. Desta forma, por meio de sua participação ativa, o usuário proporciona contribuições efetivas em todas as fases do processo de desenvolvimento, que refletem suas perspectivas e necessidades, não somente da etapa de testes e avaliação (ROCHA; BARANAUSKAS, 2003).

De acordo com Brown (1996), o DP não defende uma metodologia em particular. O argumento é de que, do ponto de vista dos defensores do DP, para a fabricação de um bom produto de *software* não é necessária uma metodologia como foco, e sim a qualidade da comunicação entre o usuário e os projetistas. No DP são definidos os tipos de comunicação desejados e métodos que podem auxiliar usuários e projetistas a realizarem um projeto melhor.

Embora uma metodologia não seja enfatizada, no DP existem métodos que são vistos mais como um recurso para os projetistas usarem como julgarem mais apropriado (BROWN, 1996).

Esses métodos se caracterizam pelo uso de técnicas simples e pelo pouco comprometimento com recursos. As técnicas mais utilizadas são: *brainstorming*, *storyboarding* e *workshops* (ROCHA; BARANAUSKAS, 2003).

3.3.5 Customização de um modelo de ciclo de vida destinada a projeto de interface para dispositivos móveis

À medida que as tecnologias móveis começam a fazer parte da infraestrutura de Tecnologia da Informação (TI) das organizações e do dia a dia das pessoas, é altamente importante, ao se construir uma estratégia de desenvolvimento de aplicações móveis, ter um entendimento aprofundado da interação do usuário com os dispositivos móveis (SOUZA; COSTA; SPINOLA, 2006).

> Novas plataformas de computação e comunicação criam a possibilidade para novos modelos de negócios e novas aplicações que têm influência na vida das pessoas. No entanto, novas plataformas também exigem a revisão dos métodos e princípios de projetos da atualidade. Os dispositivos móveis estão abrindo novas oportunidades de negócios para empresas e usuários e novos desafios de projeto de *software* (HOLTZBLATT, 2005 *apud* SOUZA; COSTA; SPINOLA, 2006, p. 7).

Os desafios tecnológicos em desenvolvimento de sistemas móveis são significativos, e o avanço da computação móvel fez surgir novos modelos de interface e exibições de informação. A interface com o usuário já tinha grande importância no processo de desenvolvimento de sistemas interativos; agora, nos projetos de *software* para computação móvel, a interface é um desafio ainda maior.

Diante desse cenário, verifica-se a grande importância de se facilitar a forma de utilização desses dispositivos computacionais móveis, isto é, de disponibilizar uma interface bem-planejada, visando facilidade de aprendizagem, simplicidade de uso e satisfação do usuário ao interagir com o sistema.

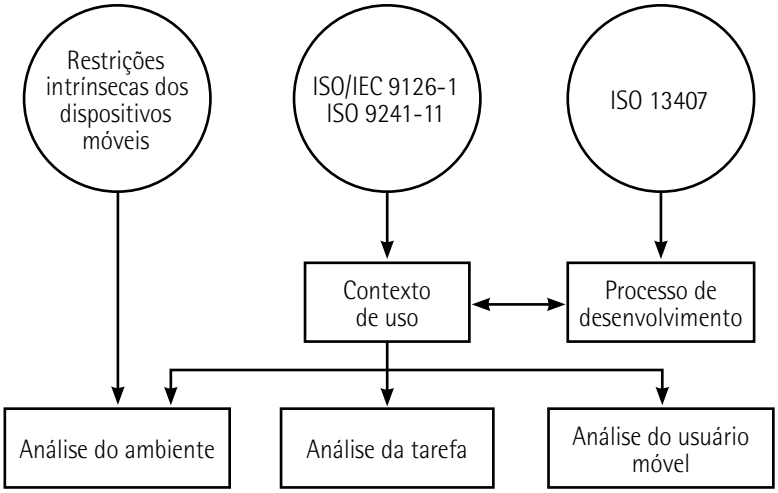

Figura 10 – Modelo de processo de desenvolvimento de aplicações móveis

A figura representa o modelo de processo de desenvolvimento de aplicações móveis. O modelo tem como base o projeto centrado no usuário, fundamentado na norma *ISO 13407: Processo de Projeto Centrado no Usuário para Sistemas Interativos*. A aplicação da norma em conjunto com a ISO 9241-11 e a ISO 9126-1 tem o objetivo de resultar em interfaces de usuário de *software* de dispositivos móveis com níveis maiores de usabilidade.

Entretanto, a diversidade e as restrições intrínsecas dos dispositivos móveis têm aumentado de maneira significativa a dificuldade e a complexidade do projeto de interface de usuário. A compreensão correta do contexto de uso móvel, ou seja, identificar e compreender as características dos usuários e seus diferentes níveis de conhecimento, o ambiente de uso e as tarefas que serão executadas com o produto desde as fases iniciais do processo de desenvolvimento, é um fator determinante na usabilidade do produto final e fundamental para o sucesso de uma aplicação móvel.

4 A USABILIDADE E AS NORMAS

Uma interface bem-projetada deve ser a mais amigável possível, possibilitando ao usuário extrair todo o poder computacional de uma aplicação e utilizá-la de forma confortável, proporcionando uma interação homem-computador transparente. Entretanto, uma interface malprojetada pode se transformar em um ponto decisivo na rejeição de um sistema, independentemente de sua funcionalidade, podendo provocar, ainda, a falha de uma aplicação que tenha sido bem-projetada e desenvolvida (SOUZA; COSTA; SPINOLA, 2006).

Algumas normas NBR e ISO estabelecem definições, orientações e recomendações sobre usabilidade que visam à produção de sistemas interativos com interfaces de usuário com qualidade.

4.1 A usabilidade e a NBR ISO 9241–11

Equivalente à norma ISO 9241-11, de 1998, a norma NBR ISO 9241-11, de 2002, sobre *Requisitos Ergonômicos para Trabalho de Escritórios com Computadores*, consiste de dezessete partes que abordam diferentes aspectos referentes ao ambiente de trabalho e a práticas do projeto de diálogo utilizado.

Para a norma (ABNT, 2002, p. 3), "o objetivo de projetar e avaliar computadores buscando usabilidade é proporcionar que usuários alcancem seus objetivos e satisfaçam suas necessidades em um contexto particular de uso". Desse modo, em sua Parte 11, "Orientações sobre usabilidade", esclarece os benefícios de medir usabilidade em termos de desempenho e satisfação do usuário e define usabilidade como: "medida na qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso", em que:

- eficácia é definida como a acurácia e a completude com as quais usuários alcançam objetivos específicos;
- eficiência é definida como os recursos gastos em relação à acurácia e à abrangência com as quais os usuários atingem os objetivos;
- satisfação é definida como a ausência do desconforto e às atitudes positivas para com o uso de um produto;
- contexto de uso é definido como usuários, tarefas, equipamentos (*hardware*, *software* e materiais), e os ambientes físico e social nos quais o produto é usado.

Observação

Embora a norma NBR 9241-11 se aplique ao trabalho de escritório com computadores, ela também pode ser aplicada a outras situações nas quais o usuário está interagindo com um produto para alcançar seus objetivos (ABNT, 2002, p. 2), por exemplo, *smartphones* e *tablets*.

A figura a seguir ilustra a estrutura de usabilidade apresentada por esta norma.

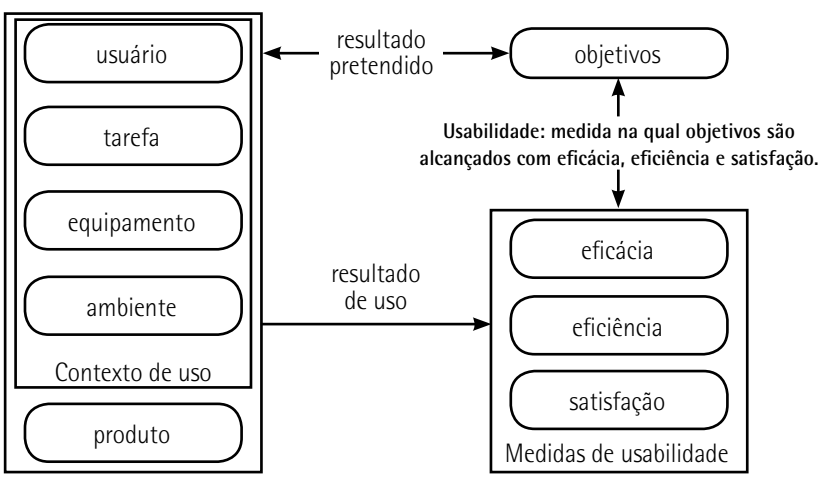

Figura 11 – Estrutura de usabilidade

Segundo a norma, a partir da identificação dos objetivos e da decomposição de eficácia, eficiência, satisfação e componentes do contexto de uso em subcomponentes com atributos

mensuráveis e verificáveis, é possível especificar ou medir a usabilidade. Para tanto, são necessárias as seguintes informações:

- uma descrição dos objetivos pretendidos;
- uma descrição (suficientemente detalhada, de modo que aqueles aspectos que possam ter uma influência significativa sobre a usabilidade possam ser reproduzidos) dos componentes do contexto de uso (existente ou uma especificação dos contextos pretendidos), incluindo usuários, tarefas, equipamento e ambientes;
- valores reais ou desejados de eficácia, eficiência e satisfação para os contextos pretendidos.

A definição de usabilidade apresentada pela NBR ISO 9241-11 também depende das qualidades de *software* – que são distintas da usabilidade – definidas na ISO/IEC 9126, tais como funcionalidade, confiabilidade e eficiência do computador. Embora, de fato, estas qualidades de *software* contribuam para a qualidade do sistema de trabalho em uso, a norma NBR ISO 9241-11 defende uma abordagem ampla na qual se encontram as necessidades de usuários reais desenvolvendo tarefas reais em um ambiente organizacional, técnico e fisicamente real. Para a norma, "a usabilidade definida em termos de qualidade de um sistema de trabalho em uso depende, necessariamente, de todos os fatores que podem influenciar no uso de um produto do mundo real" (ABNT, 2002, p. 19). Fatores organizacionais, como as práticas de trabalho e a localização ou a aparência de um produto, bem como diferenças individuais entre usuários do produto, incluindo aquelas decorrentes de fatores culturais e preferências, são elementos citados pela norma que podem influenciar a usabilidade de um produto (ABNT, 2002, p. 19).

4.2 A usabilidade e a NBR ISO/IEC 9126–1

Equivalente à norma ISO/IEC 9621-1, de 2001, a norma NBR ISO/IEC 9126-1, de 2003, *Engenharia de Software: Qualidade de Produto* é uma norma que descreve um modelo de qualidade do produto de *software* (ABNT, 2003, p. 2).

Esta norma categoriza os atributos de qualidade de *software* em seis características (funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade). Cada característica é, ainda, subdividida em subcaracterísticas, que, por sua vez, podem ser medidas por meio de métricas externas e internas.

Observação

Exemplos de métricas internas e externas são apresentados na ISO/IEC 9126-3 e na ISO/IEC 9126-2, respectivamente.

Usabilidade, para essa norma, é um atributo de qualidade de *software*, sendo apresentada como a "capacidade do produto de *software* de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas" (ABNT, 2003, p. 7). É, ainda, subdividida em cinco subcaracterísticas:

> 6.3.1 Inteligibilidade
>
> Capacidade do produto de *software* de possibilitar ao usuário compreender se o *software* é apropriado e como ele pode ser usado para tarefas e condições de uso específicas. [...]
>
> 6.3.2 Apreensibilidade
>
> Capacidade do produto de *software* de possibilitar ao usuário aprender sua aplicação. [...]
>
> 6.3.3 Operacionalidade
>
> Capacidade do produto de *software* de possibilitar ao usuário operá-lo e controlá-lo. [...]
>
> 6.3.4 Atratividade
>
> Capacidade do produto de software de ser atraente ao usuário.
>
> NOTA – Isto se refere a atributos de *software* que possuem a intenção de tornar o *software* mais atraente para o usuário, como o uso de cores e a natureza do projeto gráfico.
>
> 6.3.5 Conformidade relacionada à usabilidade
>
> Capacidade do produto de *software* de estar de acordo com normas, convenções, guias de estilo ou regulamentações relacionadas à usabilidade (ABNT, 2003, p. 9-10).

Esta norma traz ainda o conceito de **qualidade em uso** que é definido como "capacidade do produto de *software* de permitir que usuários especificados atinjam metas especificadas com eficácia, produtividade, segurança e satisfação em contextos de uso especificados" (ABNT, 2003, p. 12) e categoriza seus atributos em quatro características: eficácia, segurança, produtividade e satisfação, conforme apresentado a seguir:

> 7.1.1 Eficácia
>
> Capacidade do produto de *software* de permitir que usuários atinjam metas especificadas com acurácia e completude, em um contexto de uso especificado.
>
> 7.1.2 Produtividade
>
> Capacidade do produto de *software* de permitir que seus usuários empreguem quantidade apropriada de recursos em relação à eficácia obtida, em um contexto de uso especificado. [...]

7.1.3 Segurança

Capacidade de um produto de *software* de apresentar níveis aceitáveis de riscos de danos a pessoas, negócios, *software*, propriedades ou ao ambiente, em um contexto de uso especificado. [...]

7.1.4 Satisfação

Capacidade de um produto de *software* de satisfazer usuários, em um contexto de uso especificado (ABNT, 2003, p. 12).

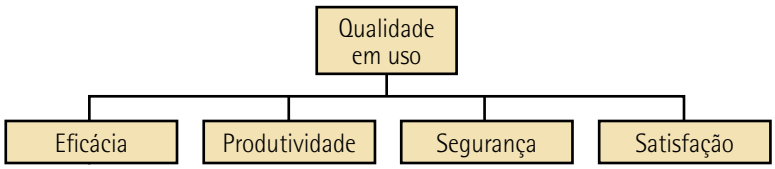

Figura 12 – Modelo de qualidade para qualidade em uso

Embora seja mais ampla, esta definição é similar à definição de usabilidade da NBR 9241-11 e também faz referência ao contexto de uso. A norma ressalta que a qualidade em uso é sob a perspectiva do usuário e é medida em termos de resultado de uso do *software* no ambiente especificado, e não em função das propriedades do próprio *software* (SOUZA; COSTA; SPINOLA, 2006).

O relacionamento entre a NBR ISO 9241-11 e a NBR ISO/IEC 9126-1 mostra que os requisitos de usabilidade para um produto estão relacionados com o contexto de uso, dependendo, portanto, do usuário, das tarefas e do ambiente.

Segundo Bevan (2001), as definições de usabilidade abordadas pelas duas normas são complementares e precisam ser combinadas durante o processo de projeto de desenvolvimento.

4.3 A norma ISO 13407 (*Processo de Projeto Centrado no Usuário para Sistemas Interativos*)

O propósito de projetar um sistema interativo é satisfazer as necessidades de usuários, ou seja, prover qualidade em uso (BEVAN, 1999), que é (ou pelo menos deveria ser) o meio para alcançar qualidade nos produtos de *software* (BEVAN; BOGOMOLNI, 2000).

A norma ISO 13407, de 1999, sobre *Processo de Projeto Centrado no Usuário para Sistemas Interativos* fornece orientações sobre as atividades de projeto centrado no usuário que acontecem ao longo do ciclo de vida de sistemas interativos computacionais (ISO, 1999). Descreve um ciclo de desenvolvimento iterativo no qual as especificações de requisitos de produto esclarecem corretamente os requisitos do usuário e da organização, bem como especificam o contexto no qual o produto será usado.

Essa norma define um conjunto de princípios que incorporam a perspectiva do usuário no processo de desenvolvimento de *software*.

- Distribuição apropriada de função entre usuário e sistema: determina quais aspectos do trabalho ou da tarefa devem ser controlados por *software* e *hardware*.

- Envolvimento ativo de usuários: utiliza as pessoas que têm maiores conhecimentos no contexto em que a aplicação será usada, visando, com isso, a um aumento no compromisso de participação no desenvolvimento do *software*.

- Repetição de soluções de projeto: requer a avaliação contínua nas fases iniciais dos usuários finais por técnicas de prototipação diferentes.

- Equipes multidisciplinares: alimentam um processo de desenvolvimento colaborativo com o envolvimento de especialistas de várias áreas, cada um cooperando e compartilhando seus conhecimentos.

De acordo com a norma, há quatro principais atividades, apresentadas na figura a seguir, que devem ser empregadas para incorporar requisitos de usabilidade ao processo de desenvolvimento de *software* centrado no usuário.

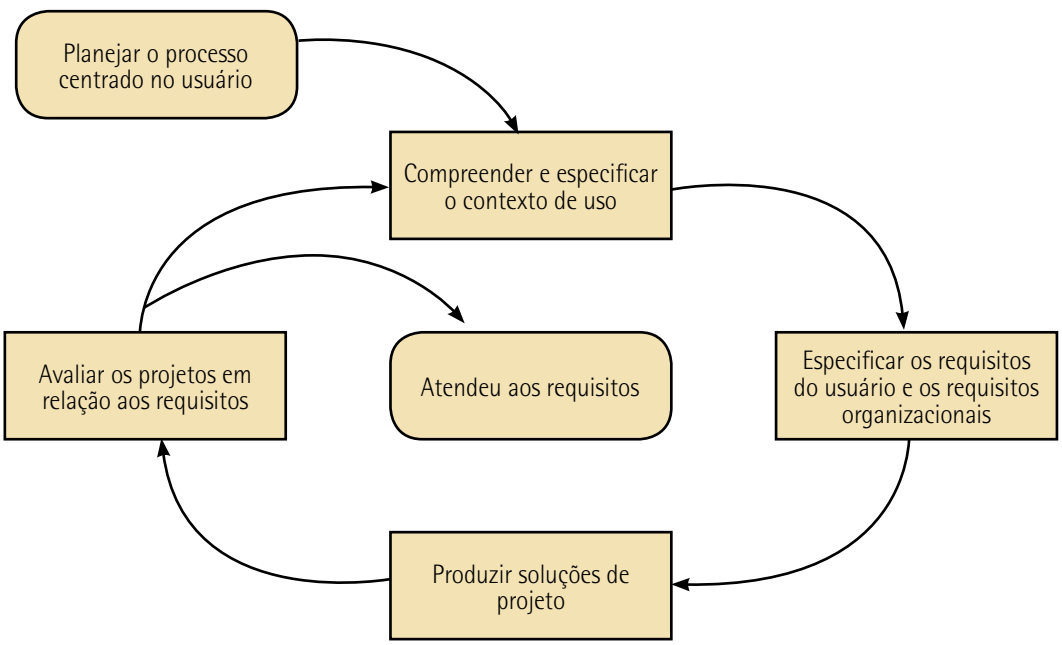

Figura 13 – Processo de projeto centrado no usuário

- Compreender e especificar o contexto de uso: o objetivo é obter as informações sobre as características dos usuários, o ambiente de uso e as tarefas que serão executadas com o produto, além de fornecer uma base para as atividades de avaliações posteriores.

- Especificar os requisitos do usuário e da organização: determinar os critérios de sucesso para a usabilidade do produto em termos de tarefas realizadas pelos usuários, bem como diretrizes e limitações do projeto.

- Produzir soluções de projeto: incorporar conhecimentos de interface homem-computador às soluções de projeto. As possíveis soluções de projeto são exploradas, sendo descritas por meio da utilização de protótipos. As primeiras soluções de projeto podem ser baseadas em experiências anteriores ou utilização de normas e guias, que são refinadas por *feedback* do usuário.

- Avaliar projetos em relação aos requisitos do usuário: a usabilidade do projeto deve ser avaliada em relação às tarefas dos usuários, tendo como objetivo confirmar o nível em que os requisitos da organização e dos usuários foram alcançados, fornecendo também informações para o refinamento do projeto.

O ciclo dessas atividades termina quando a "avaliação do projeto em relação aos requisitos do usuário" é executada com um resultado satisfatório.

Os benefícios da usabilidade através do projeto centrado no usuário orientados pela norma ISO 13407 podem incluir o aumento da produtividade, aumento na qualidade de trabalho, reduções de custos em treinamento e aumento da satisfação do usuário.

O relatório técnico ISO TR 18529, de 2000, sobre *Ergonomia de Interação Homem-Sistema - Descrições sobre o Ciclo de Vida Centrado no Usuário*, contém uma definição e uma estrutura formalizada do processo centrado no usuário descrito na ISO 13407. O modelo de maturidade de usabilidade descrito na ISO TR 18529 pode ser utilizado em conjunto com a ISO 15504, *Avaliação de Processo de Software*, para avaliar a capacidade de uma organização na utilização do processo de desenvolvimento centrado no usuário (ISO STANDARDS, 2006).

Fonte: Souza, Costa e Spinola (2006, p. 5-7).

Saiba mais

Diversas normas internacionais foram desenvolvidas para definir os princípios gerais de projeto centrado no usuário. Um estudo mais aprofundado sobre aplicação das normas ISO pode ser encontrado em:

BEVAN, N. International standards for HCI and usability. *International Journal of Human Computer Studies*, Londres, v. 55, n. 4, p. 533-52, out. 2001. Disponível em: <http://www.nigelbevan.com/papers/HCI-Usability_standards.pdf>. Acesso em: 2 mar. 2015.

Resumo

Vimos que a Engenharia de *Software* tem o foco voltado para o produto e seu processo de desenvolvimento, enquanto a Interação Humano-Computador tem o foco voltado à interação do ser humano e da máquina. No entanto, as duas áreas estabelecem métodos e processos de desenvolvimento de sistemas interativos. Enquanto a Engenharia de *Software* tem o foco no projeto e na especificação da funcionalidade interna do sistema, bem como em sua arquitetura, visando à qualidade estrutural do produto de *software* final, a IHC focaliza a interação e o *design* de interface do usuário, levando em conta as necessidades, os valores e as expectativas dos usuários, visando à qualidade de uso da solução projetada. Portanto, ambas as áreas tratam da qualidade do produto final, no entanto sob perspectivas e focos diferentes.

Abordamos que a Interação Humano-Computador tornou-se um componente essencial para todos os profissionais de computação. Desenvolvedores, projetistas e engenheiros de *software* também precisam entender os princípios e conceitos de Interação Humano-Computador; ainda que não sejam os principais profissionais responsáveis pela compreensão do usuário e pelo projeto da interface, trabalharão com os profissionais responsáveis.

Aprendermos que a interface de usuário vem ganhando ainda mais importância nos projetos de sistemas interativos. Entretanto, a inclusão do usuário no desenvolvimento de interfaces, embora já não seja uma fronteira tão nova assim na Engenharia de *Software*, ainda não acontece consoante sua importância.

Durante os estudos, vimos que a Engenharia de *Software* propõe vários modelos de ciclo de vida de *software*. As fases tipicamente utilizadas neles, aplicáveis à grande maioria dos projetos de *software*, são: Requisitos, Análise, Projeto (*Design*), Implementação, Testes e Implantação. Estas fases podem se sobrepor ou podem ser executadas iterativamente. Os principais e mais tradicionais modelos de ciclo de vida de *software* são: cascata, incremental, espiral e prototipagem.

Abordamos ainda que a prototipagem (ou prototipação) permite que todo o sistema, ou parte dele, seja construído rapidamente para que questões sejam entendidas ou esclarecidas. O paradigma de prototipagem pode ser dividido em evolucionário e descartável (*throw-away*). Na prototipagem evolucionária, a equipe de desenvolvimento trabalha com

os *stakeholders* os aspectos mais visíveis do sistema, normalmente a interface de usuário, sendo muito especialmente útil quando os usuários possuem dificuldade em expressar os requisitos do sistema. A prototipagem descartável (*throw-away*) consiste na construção de protótipos utilizados para explorar as alternativas de projeto, entender melhor os requisitos do sistema e reduzir riscos. O protótipo gerado nesse modelo não tem o objetivo de ser um sistema funcional. Depois de cumprir com seu propósito, ele será descartado.

Vimos que a IHC também apresenta modelos de ciclo de vida para o desenvolvimento de sistemas interativos. No entanto, os modelos de ciclo de vida apresentados pela IHC não discutem as etapas para o desenvolvimento e a implementação do sistema e suas funcionalidades, diferentemente dos modelos de ciclo de vida da Engenharia de *Software*. Esses modelos enfocam o desenvolvimento da interação com o usuário. Os principais são os Modelos Estrela, Engenharia de Usabilidade, Projeto Centrado no Usuário e *Design* participativo.

Tivemos a oportunidade de estudar e aprender sobre a importância das normas no processo de desenvolvimento de interface de usuário. Algumas normas NBR e ISO estabelecem definições, orientações e recomendações sobre usabilidade que visam à produção de sistemas interativos com interfaces de usuário com qualidade. De acordo com a NBR ISO 9241-11, sobre *Requisitos Ergonômicos para Trabalho de Escritórios com Computadores*, a partir da identificação dos objetivos e da decomposição de eficácia, eficiência, satisfação e componentes do contexto de uso em subcomponentes com atributos mensuráveis e verificáveis, é possível especificar ou medir usabilidade. A definição de usabilidade apresentada pela NBR ISO 9241-11 também depende das qualidades de *software*, que são distintas daquelas definidas na ISO/IEC 9126, tais como funcionalidade, confiabilidade e eficiência do computador. A NBR ISO/IEC 9126-1, *Engenharia de Software: Qualidade de Produto*, é uma norma que descreve um modelo de qualidade do produto de *software*. O relacionamento entre as normas NBR ISO 9241-11 e NBR ISO/IEC 9126-1 mostra que os requisitos de usabilidade para um produto estão relacionados com o contexto de uso, dependendo, portanto, do usuário, das tarefas e do ambiente.

A norma ISO 13407, sobre *Processo de Projeto Centrado no Usuário para Sistemas Interativos*, fornece orientações sobre as atividades de projeto centrado no usuário que acontecem ao longo do ciclo de vida de sistemas interativos computacionais. Os benefícios da usabilidade por meio do projeto centrado no usuário orientados pela norma ISO 13407 podem incluir o aumento da produtividade, aumento na qualidade de trabalho, reduções de custos em treinamento e aumento da satisfação do usuário.

Exercícios

Questão 1. (Enade 2008) Considere que você trabalhe em uma empresa de desenvolvimento de *software* e que a empresa tenha decidido desenvolver um novo editor de texto para colocar no mercado. Esse editor deve ser um *software* que forneça recursos adicionais de apoio à autoria, embasado no estilo de escrita do usuário, o que o torna um *software* de funcionalidade mais complexa. Considere que a empresa deseje disponibilizar o produto no mercado em versões que agreguem esse suporte de forma gradativa, fazendo análise de risco para avaliar a viabilidade de desenvolvimento de uma nova versão. Tendo de escolher um modelo de processo para desenvolver esse editor, e conhecendo as características dos modelos existentes, entre os modelos a seguir, qual é o modelo apropriado para esse caso?

A) Cascata.

B) Espiral.

C) RAD (Rapid Application Development).

D) Prototipação.

E) Cleanroom.

Resposta correta: alternativa B.

Análise das alternativas

A) Alternativa incorreta.

Justificativa: no modelo cascata as atividades são realizadas de forma sequencial em um único ciclo de desenvolvimento, contrariando o enunciado, que pede um modelo incremental com vários ciclos de desenvolvimento.

B) Alternativa correta.

Justificativa: o modelo espiral permite a condição de incremento, com vários ciclos de desenvolvimento, e agrega a característica de análise de riscos.

C) Alternativa incorreta.

Justificativa: no modelo RAD há o incremento das atividades, mas a análise de riscos não está presente.

D) Alternativa incorreta.

Justificativa: na prototipação há o incremento das atividades, mas não existe a análise de riscos. O protótipo é constantemente validado pelo cliente.

E) Alternativa incorreta.

Justificativa: o objetivo do cleanroom é entregar o *software* com o menor número de erros possível. São realizados testes estatísticos e efetuadas a especificação formal e a verificação estática por inspeções. O processo é incremental, mas não contempla a análise de riscos.

Questão 2. (Enade 2011) O levantamento de requisitos é uma etapa fundamental do projeto de sistemas. Dependendo da situação encontrada, uma ou mais técnicas podem ser utilizadas para a elicitação dos requisitos. A respeito dessas técnicas, analise as afirmativas a seguir.

I – *Workshop* de requisitos consiste na realização de reuniões estruturadas e delimitadas entre os analistas de requisitos do projeto e representantes do cliente.

II – Cenário consiste na observação das ações do funcionário na realização de uma determinada tarefa, para verificar os passos necessários para sua conclusão.

III – As entrevistas são realizadas com os *stakeholders* e podem ser abertas ou fechadas.

IV – A prototipagem é uma versão inicial do sistema, baseado em requisitos levantados em outros sistemas da organização.

É correto apenas o que se afirma em:

A) I e II.

B) I e III.

C) II e IV.

D) I, III e IV.

E) II, III e IV.

Resolução desta questão na plataforma.