Unidade III

7 INTERFACE HOMEM-COMPUTADOR, QUALIDADE E AVALIAÇÃO

7.1 Princípios da Interação Humano-Computador (IHC)

Frustração e ansiedade são partes da vida diária para muitos usuários de sistemas de informação computadorizados. A figura 46 ilustra bem essa situação em que os usuários lutam para entender um menu de opções que se destinam a ajudá-los. Algumas pessoas enfrentam casos sérios de susto com computador ou neurose, que os levam a evitar sistemas computadorizados.

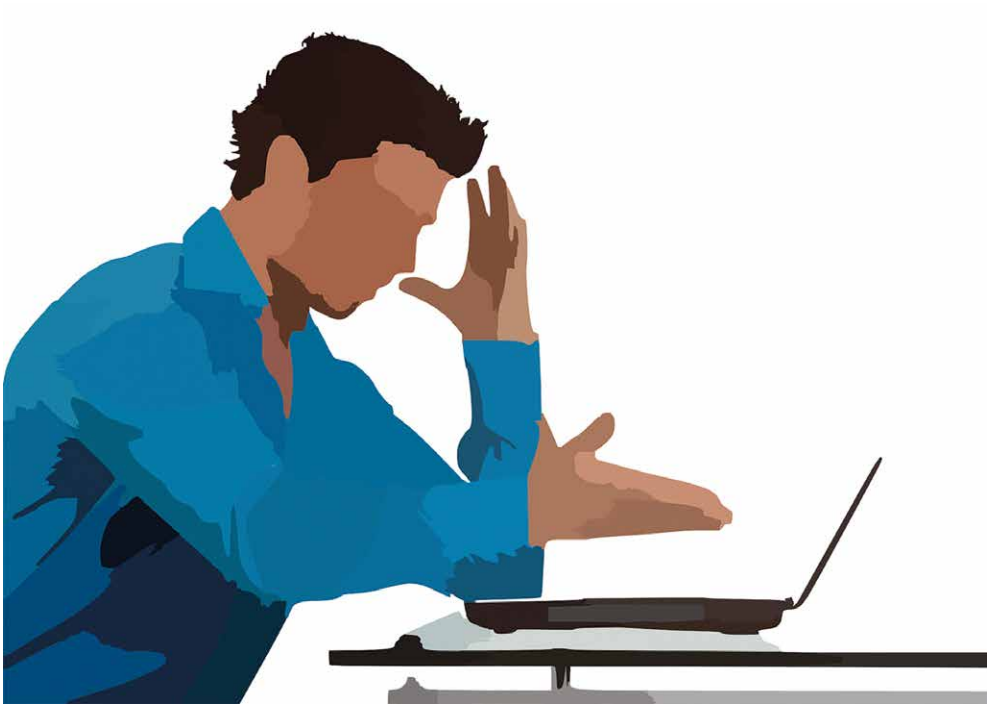

Figura 46 – Humano frustrado com o computador

Disponível em: http://tinyurl.com/49ecyzhn. Acesso em: 19 jan. 2024.

7.1.1 Definição de interface e interação

Segundo o dicionário Houaiss, interface é o "elemento que proporciona uma ligação física ou lógica entre dois sistemas ou partes de um sistema que não poderiam ser conectados diretamente" (2015).

Veja a figura 47, na computação a interface pode ser definida como a parte de um sistema computacional com a qual a pessoa entra em contato: físico, perceptivo ou conceitualmente. Por exemplo,

superfícies que separam duas camadas quaisquer, teclado para entrada de dados no computador pelo ser humano e o vídeo para visualizar dados do computador.

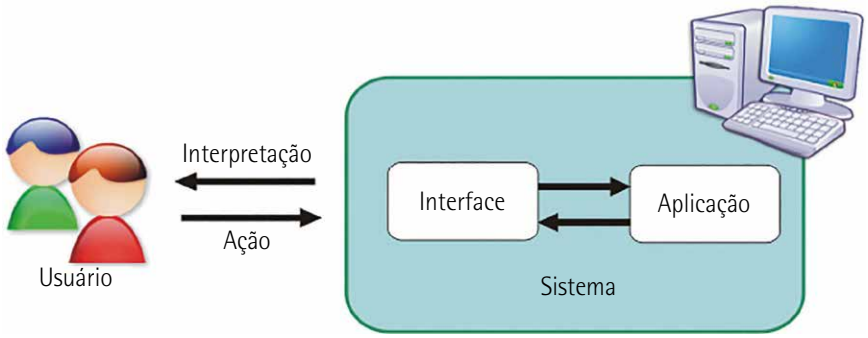

Figura 47 – Comunicação entre o usuário e o sistema feita pela interface

Fonte: Fortunato (2020).

De acordo com o mesmo dicionário, interação é a "atividade ou trabalho compartilhado, em que existem trocas e influências recíprocas" (2015). Veja a figura 48, na computação o termo interação pode ser definido como o processo de comunicação entre pessoas e sistemas interativos. Por exemplo, ação recíproca de dois ou mais corpos, execução de um comando pelo computador, bem como o resultado fornecido pelo computador.

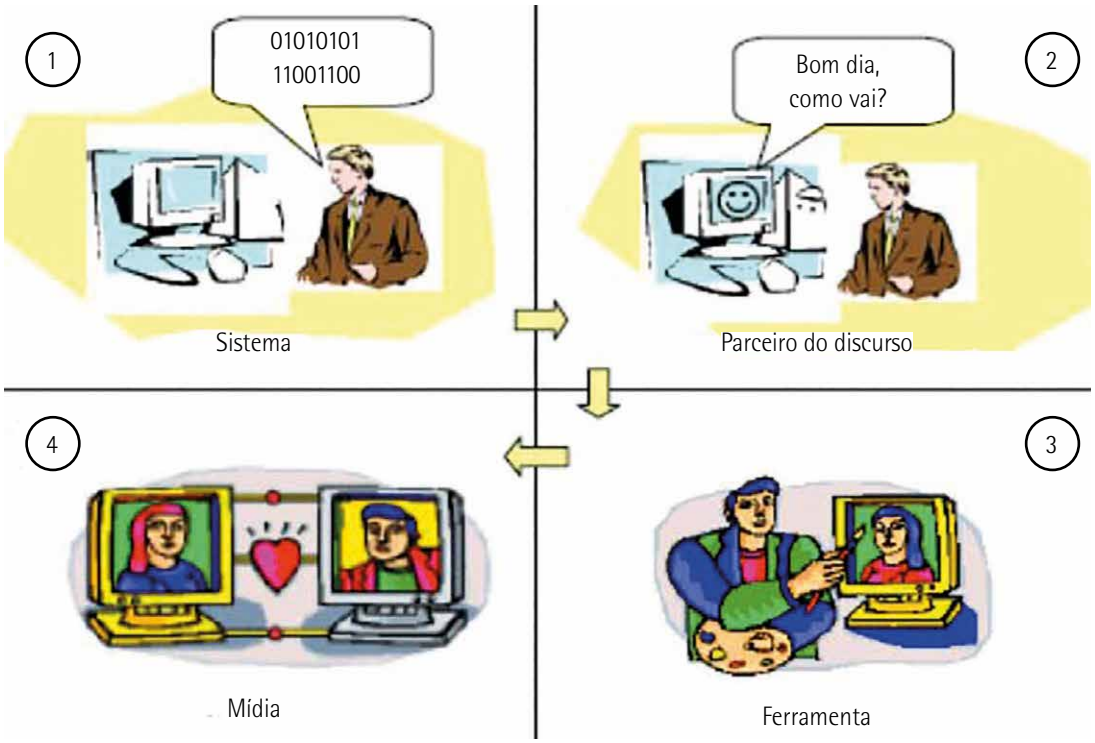

Figura 48 – Quatro perspectivas de interação humana com o computador

Fonte: Carvalho (2015).

O objetivo da área de interação humano-computador é fornecer explicações e previsões para fenômenos de interação do usuário com o sistema e resultados práticos para o design da interface de usuário (ACM SIGCHI, 1992).

Como pergunta Pressman, "a maioria das limitações e restrições de interface que são impostas por um projetista são destinadas a simplificar o modo de interação. Mas para quem?" (2007). Em 1997, Mandel (*apud* Pressman, 2011) define alguns princípios do projeto de interface com o usuário, que permitem ao usuário manter o controle:

- Definir os modos de interação de forma a não forçar o usuário a ações desnecessárias ou indesejadas.
- Proporcionar interação flexível.
- Permitir que a interação com o usuário possa ser interrompida e desfeita.
- Simplificar a interação à medida que os níveis de competência progridem e permitam que a interação seja personalizada.
- Esconder detalhes técnicos internos do usuário esporádico.
- Projetar a interação direta com objetos que aparecem na tela.

7.1.2 Interface com o usuário: modos de acesso dos sistemas operacionais

De acordo com Pressman (2011), o conceito de projeto de interface com o usuário é criar um meio efetivo de comunicação entre o ser humano e o computador. Seguindo um conjunto de princípios de projeto de interface, o projeto identifica objetos e ações de interface e depois é criado um layout de tela que forma a base para um protótipo de interface com o usuário.

Os modos mais comuns de acesso aos sistemas operacionais ocorrem basicamente de três formas:

1. Interface de texto, linha de comando ou 1D (uma dimensão): basicamente o teclado e monitor de vídeo são usados para o usuário ativar o software via texto. O cursor pode se deslocar para direita ou esquerda. Exemplos: DOS (figura 49) e Linux (figura 50), no modo texto.

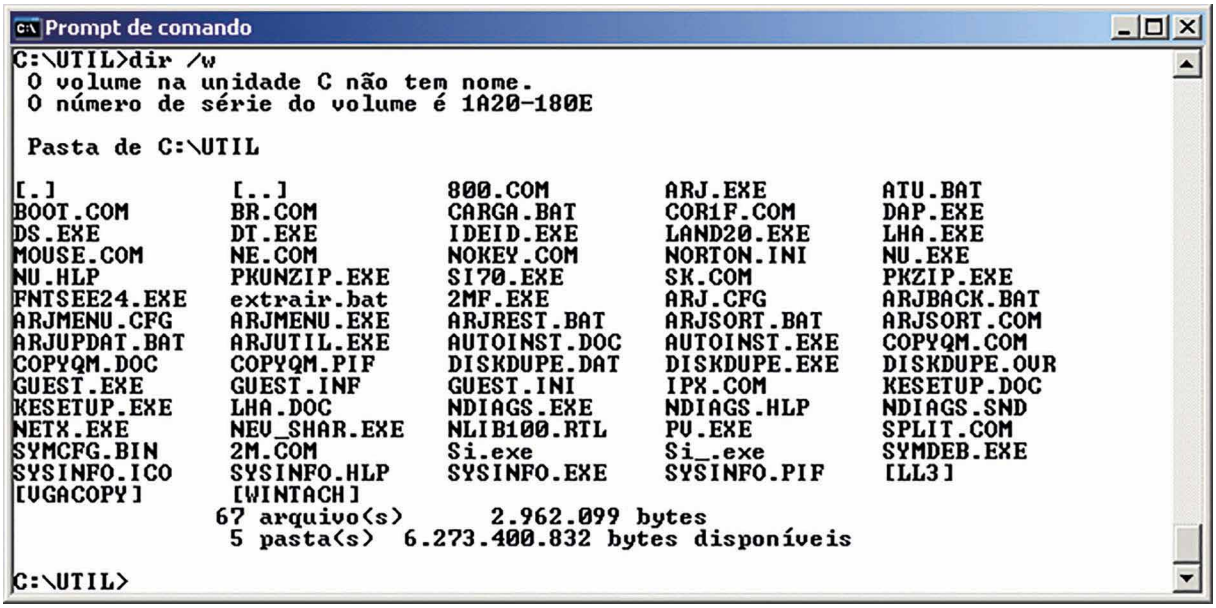

Figura 49 – Sistema operacional DOS (*Disk Operation System*)

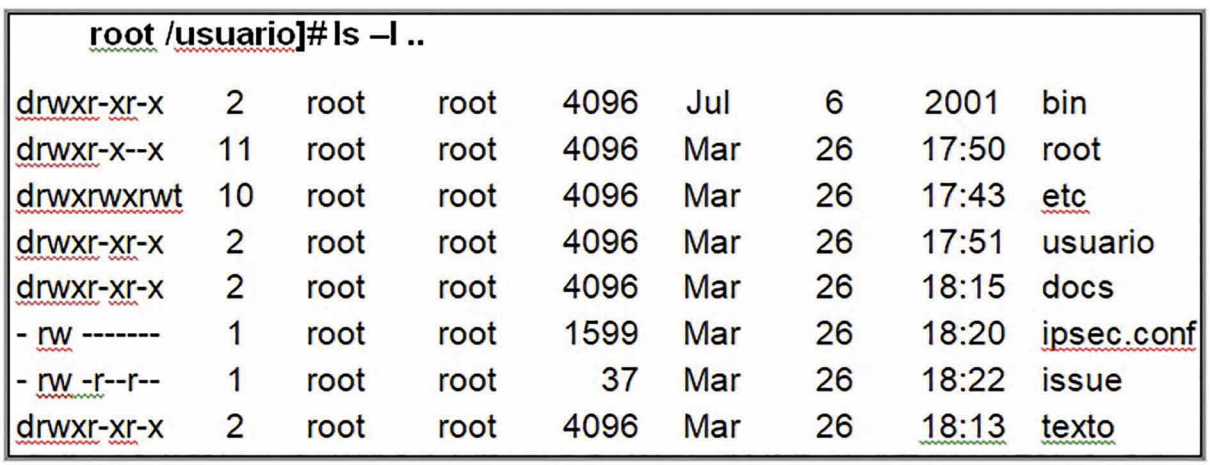

Figura 50 – Tela de exibição no modo texto do Linux, do diretório "root/usuário"

2. Interface gráfica ou 2D (duas dimensões): além do teclado e monitor de vídeo, usa-se o mouse para deslocar um ponteiro por toda a tela. Por meio de figuras (ícones) mostradas na tela, o software pode ser selecionado ou ativado (executado), pressionando os botões do mouse. Em informática, a Interface Gráfica do Usuário (GUI – *Graphical User Interface*) é um tipo de interface do usuário que permite a interação com dispositivos digitais através de elementos gráficos como ícones e outros indicadores visuais, em contraste com a interface de linha de comando. Exemplos: Windows (figura 51), Linux no modo gráfico (figura 52), MAC OS e Android.

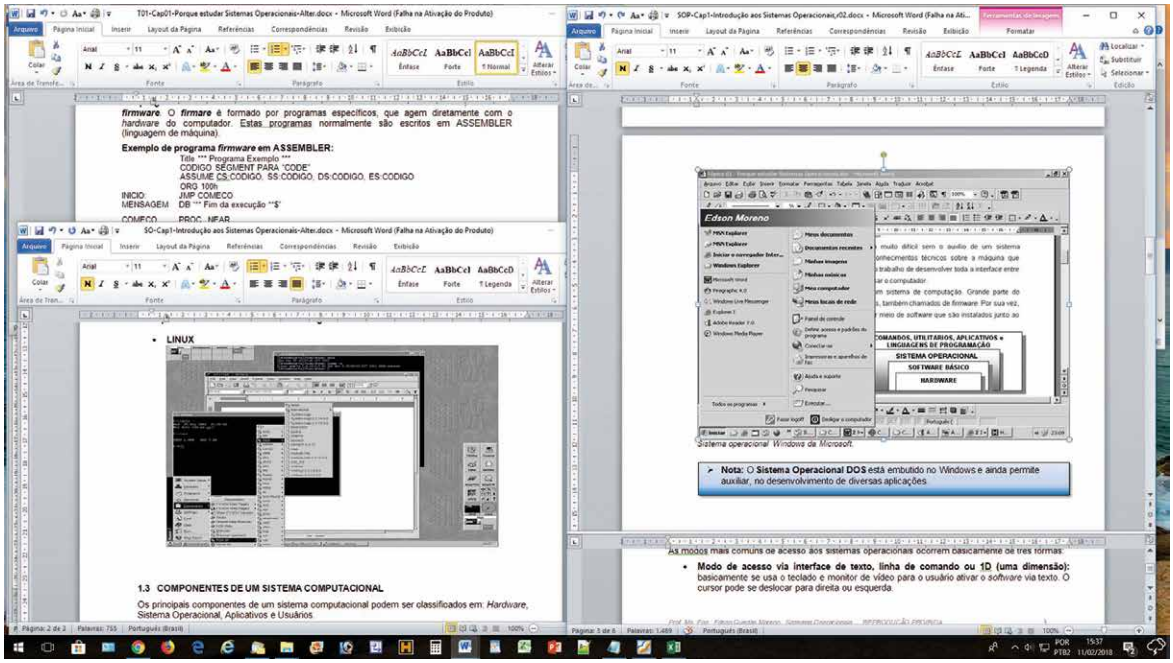

Figura 51 – Sistema operacional Windows da Microsoft

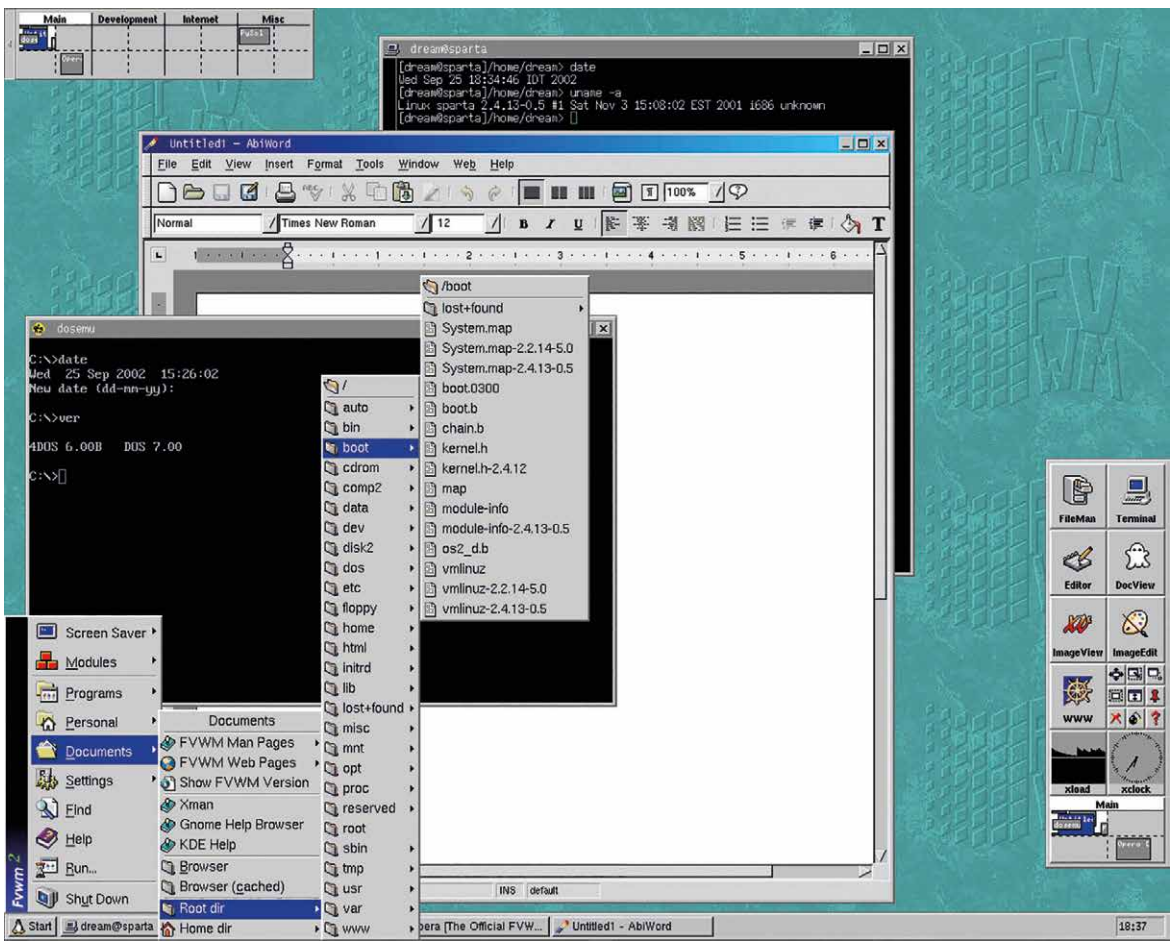

Figura 52 – Tela de exibição no modo gráfico do Linux, do diretório "root/usuário"

3. Interfaces 3D (três dimensões): este tipo de interface permite, por meio dos sentidos e características dos humanos, a comunicação com o computador. É uma interface que imita alguns aspectos do ser humano, tais como: manipulação por voz, por movimentos, pressão, movimento dos olhos e várias outras formas.

Saiba mais

Olhe o que já está por aí! A Interface Homem-Computador combinada com a Inteligência Artificial (IA). Veja as especificações técnicas e vídeo sobre o HoloLens 2:

MICROSOFT. *HoloLens 2*. [s.d.]. Disponível em: http://tinyurl.com/yv3jpf4p. Acesso em: 18 nov. 2023.

7.1.3 Qualidade em uso: usabilidade, comunicabilidade e acessibilidade

A Qualidade em Uso visa não apenas cobrir a facilidade de uso, mas também as funcionalidades e o suporte apropriado às atividades de uso em cenário real. É considerada não somente a visão do usuário, mas o contexto de uso em ambiente de trabalho (Pagani, 2011).

Qualidade em Uso é um termo cunhado em meados da década de 1990 por Nigel Bevan e incorporado à norma ISO/IEC 9126, originalmente se referindo à qualidade de software. Ela estende a caracterização da qualidade e ergonomia de software contida em determinadas normas ISO, englobando o contexto do ambiente de trabalho para caracterizar a satisfação de uso, focando não apenas no usuário, mas em seu comportamento ao interagir com um sistema computacional. A qualidade interna (propriedades do código) e a qualidade externa (comportamento do software durante a execução) influenciam a qualidade em uso e estão contidas nela. Dessa forma, temos a percepção da qualidade verificada através do uso (Pagani, 2011).

- Critérios de qualidade de uso

  – Usabilidade: Pressman (2011) define usabilidade como a "capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições específicas". Veja o exemplo a seguir (figura 53): uma característica que permite melhorar o uso do software é facilitar a navegação do usuário colocando em destaque o "botão" a ser pressionado.

Figura 53 – Usabilidade: colocar em destaque o "botão" a ser pressionado

Fonte: Krug (2006).

– Comunicabilidade: Neves (2012) define como a "propriedade de transmitir ao usuário de forma eficaz e eficiente, as intenções e princípios de interação que guiaram o seu design". Tornar o software cada vez mais aplicável aumenta as chances de o usuário utilizar o software de modo mais criativo, eficiente e produtivo. Um exemplo é a tela de apresentação do "Windows Media Player", mostrado na figura 54.

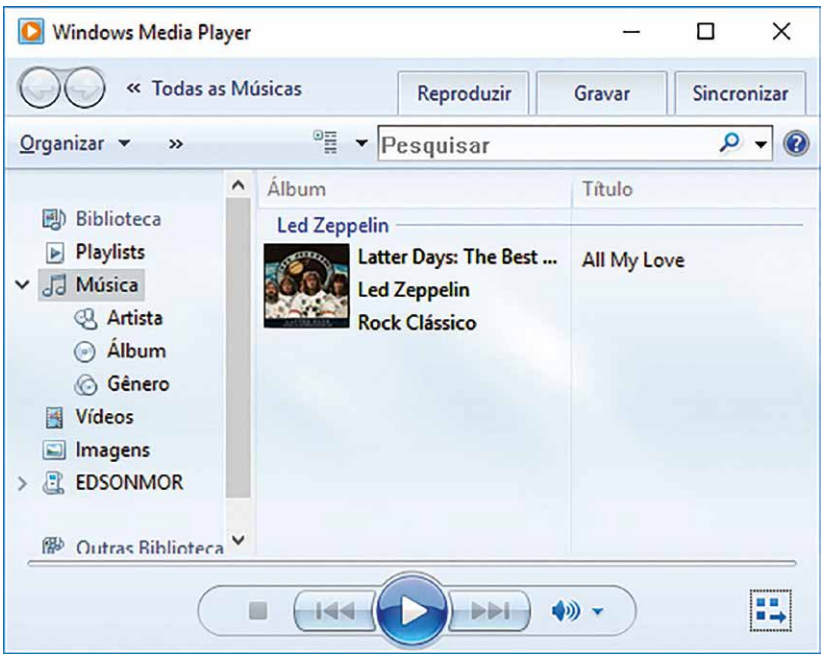

Figura 54 – Tela do software Windows Media Player

– Acessibilidade: na definição de Neves (2012), significa "remover barreiras de acesso baseado em limitações técnicas, ambientais e de deficiência". Uma abordagem sobre acessibilidade pode ser vista na representação da figura 55, que é a logomarca do Conselho Municipal dos Direitos das Pessoas com Deficiência de Pompeu (CMDPD).

Implementar características de acessibilidade permite desenvolver meios ou interfaces que facilitem a um deficiente de audição ou visão operar um software. É o caso de conversores de textos em fala ou vice-versa, para cegos.

Figura 55 – Logomarca do Conselho Municipal dos Direitos das Pessoas com Deficiência de Pompeu (CMDPD)

Fonte: CMDPD (2014).

7.1.4 Retorno de investimento (ROI)

O objetivo principal do software é definir uma boa funcionalidade e ser de fácil uso, para que o usuário possa se sentir satisfeito com a aplicação, ou seja um software bem atrativo. O retorno do investimento (do inglês *Return Over Investiment*) na IHC pode ser observado na implementação de uma boa interface humano-computador, que atenda as seguintes características:

- rápido treinamento;
- motivação à exploração ou navegação;
- termos associados ao paradigma do usuário, que não os confunda ou induzam ao erro;
- satisfação de uso;
- aumento da produtividade.

Atender a estas características melhora a qualidade de uso, aumentando o benefício em relação ao capital investido em uma boa interface.

Observação

A IHC desempenha um papel crucial na melhoria do ROI em vários contextos: usabilidade aprimorada, eficiência operacional, redução de erros, adaptabilidade em relação a diversos dispositivos, melhoria na UX e uma análise de dados mais eficiente.

7.2 Normas de qualidade: ISO 9126, ISO 14598 e ISO 25000

Os computadores têm sido usados numa variedade de áreas de aplicação cada vez maior e sua correta operação é frequentemente crítica para o sucesso de negócios e para a segurança humana. Deste modo, desenvolver ou selecionar produtos de software de alta qualidade é de primordial importância. Especificação e avaliação da qualidade do produto de software são fatores-chave para garantir qualidade adequada. Isto pode ser alcançado pela definição apropriada das características de qualidade, levando em consideração o uso pretendido do produto de software. É importante que cada característica relevante de qualidade do produto de software seja especificada e avaliada, utilizando, quando possível, métricas validadas ou amplamente aceitas (ABNT, 2003).

As características de qualidade e métricas associadas são úteis para avaliar o produto de software, requisitos de qualidade e outros usos. Para auxiliar essas necessidades, em 1991 foi criada a NBR ISO/IEC 9126 (Qualidade do produto de software), em 1996 a NBR 13596 (Tecnologia de informação – Avaliação de produto de software – Características de qualidade e diretrizes para o seu uso) e em 1998 a NBR ISO/IEC 14598 (Avaliação de produto de software).

Em um quadro evolutivo, a NBR ISO/IEC 9126 substituiu a NBR ISO/IEC 13596. Uma relação entre as NBR ISO/IEC 9126 e NBR ISO/IEC 14598 pode ser vista na figura 56:

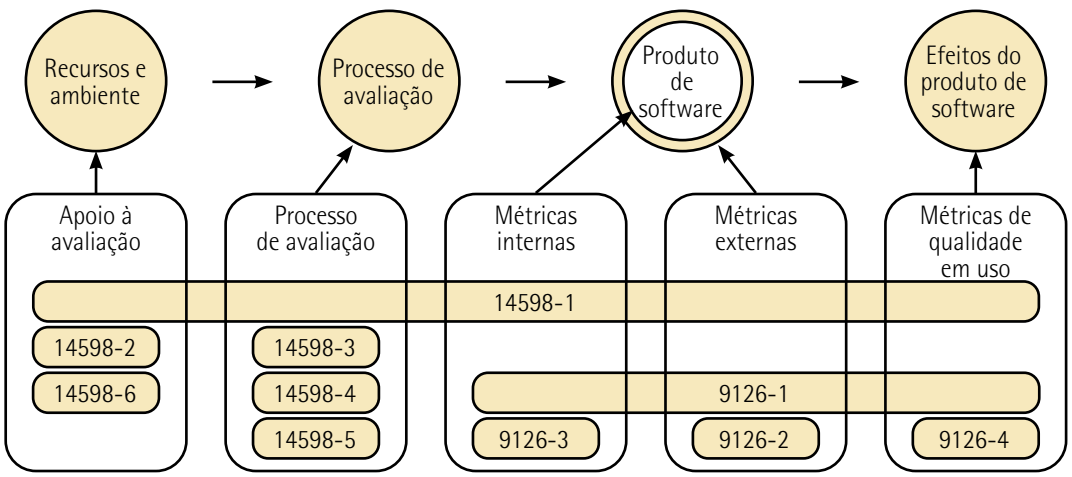

Figura 56 – Relação entre as NBR ISO/IEC 9126 e NBR ISO/IEC 14598

Fonte: ABNT (2003).

7.2.1 NBR ISO/IEC 9126 – Qualidade do produto da engenharia de software

As características de qualidade do produto de software definidas na NBR ISO/IEC 9126 podem ser usadas para especificar requisitos funcionais e não funcionais do cliente e do usuário.

A qualidade do produto é composta por um modelo de qualidade com características e subcaracterísticas (internas e externas) que podem ser usadas como parâmetros relacionados à qualidade.

Na avaliação precisam ser alocados diferentes tipos de medições dependendo dos objetivos do negócio, dos cenários de uso e do processo utilizado no projeto.

Principais usos da NBR ISO/IEC 9126:

- **Avaliação da qualidade de software**: a norma fornece um modelo de qualidade que incluía características específicas e subcaracterísticas associadas para avaliar diferentes aspectos da qualidade do software, como funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Com esse modelo é possível medir a qualidade do software e identificar áreas que precisavam de melhoria.

- **Definição de requisitos de qualidade**: permite que desenvolvedores e clientes especifiquem os requisitos de qualidade do software de maneira mais precisa, uma vez que fornece um conjunto claro de características e métricas para medir cada aspecto da qualidade.

- **Comunicação entre fornecedores e clientes**: Facilita a comunicação entre fornecedores e clientes em relação à qualidade do software. As partes podem usar o modelo de qualidade como referência para entender e negociar os requisitos de qualidade do produto.

- **Melhoria do processo de desenvolvimento**: Ao identificar as características de qualidade e medir a qualidade do software, as organizações podem utilizar os resultados para melhorar seus processos de desenvolvimento, visando entregar produtos de software de maior qualidade.

A estrutura do modelo de qualidade da NBR ISO/IEC 9126 como mostra a figura 57 explica o relacionamento entre diferentes abordagens para qualidade.

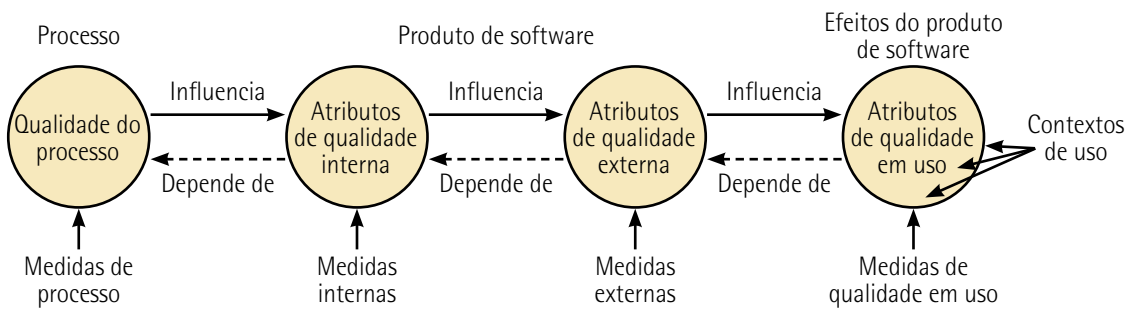

Figura 57 – Qualidade no ciclo de vida

Fonte: ABNT (2003).

Os requisitos de qualidade externa incluem requisitos derivados das necessidades de qualidade dos usuários, incluindo os requisitos de qualidade em uso, usados como meta para validação em vários estágios de desenvolvimento. Convém que os requisitos de qualidade externa, sejam declarados na especificação de requisitos de qualidade usando métricas externas.

Os requisitos de qualidade interna especificam o nível de qualidade requerido sob o ponto de vista interno do produto. Podem incluir modelos estáticos e dinâmicos, outros documentos e código-fonte. Podem ser usados também para definir estratégias de desenvolvimento e critérios de avaliação e de verificação durante o desenvolvimento, incluindo o uso de métricas adicionais (por exemplo, reusabilidade). Convém que os requisitos de qualidade interna sejam definidos quantitativamente usando métricas internas.

O modelo de qualidade para qualidade externa e interna mostrado na figura 58 categoriza os atributos de qualidade de software em seis características (funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade), por sua vez, subdivididas em subcaracterísticas que podem ser medidas por meio de métricas externas e internas.

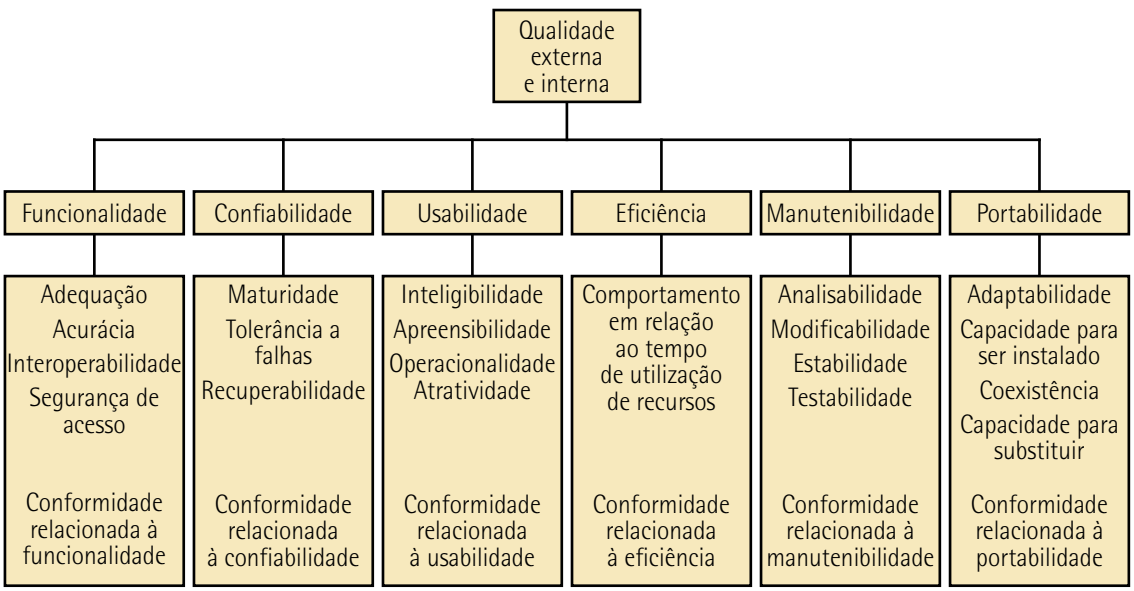

Figura 58 – Modelo de qualidade para qualidade externa e interna

Fonte: ABNT (2003).

Qualidade em uso é a visão da qualidade do produto de software do ponto de vista do usuário, quando este produto é usado em um contexto e cenário de uso específico. A qualidade em uso mede o quanto usuários podem atingir seus objetivos num determinado ambiente e não as propriedades do software.

O modelo de qualidade para qualidade em uso apresentado na figura 59 define os atributos de qualidade em uso categorizados em quatro características: eficácia, produtividade, segurança e satisfação.

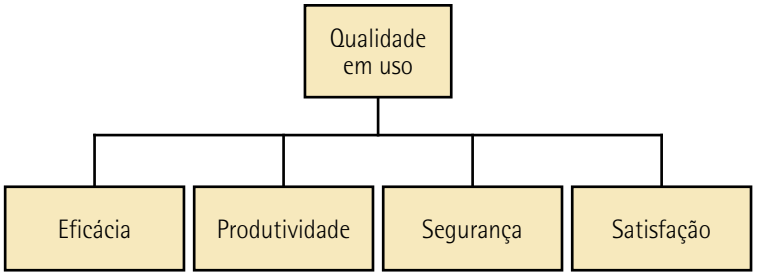

Figura 59 – Modelo de qualidade para qualidade em uso

Fonte: ABNT (2003).

Lembrete

Reveja o tópico 2.1.1 Software Development Life Cycle (SDLC). Vemos que ao integrarmos a ISO 9126 ao SDLC, os engenheiros de software podem integrar práticas de desenvolvimento do software em todo o seu ciclo de vida. As avaliações de qualidade em seus processos garantem que a qualidade seja uma consideração central em todas as fases do desenvolvimento de software.

7.2.2 NBR ISO/IEC 14598 – Avaliação do produto de software

Com o crescente uso da tecnologia de informação, cresce também o número de sistemas computacionais críticos. Estes sistemas incluem, por exemplo, sistemas críticos quanto à segurança de acesso, segurança de vidas humanas, finanças e meio ambiente. A qualidade de software desses sistemas é particularmente importante porque defeitos no software podem levar a consequências sérias (NBR ISO/IEC 14598, 2001).

A NBR ISO/IEC 14598 fornece uma visão geral das outras partes e também explica o relacionamento entre a NBR ISO/IEC 14598 e o modelo de qualidade apresentado na NBR ISO/IEC 9126.

A NBR ISO/IEC 14598 fornece uma estrutura para avaliar a qualidade de quaisquer produtos de software e estabelece os requisitos para métodos de medição e avaliação de produtos de software, define termos técnicos utilizados nas demais partes, contém requisitos gerais para especificação e avaliação da qualidade de software e esclarece os conceitos gerais.

No apoio para a avaliação cada uma das normas de processos de avaliação pode ser utilizada em conjunto com a ISO/IEC 14598-2 (Planejamento e gestão) e ISO/IEC 14598-6 (Documentação de módulos de avaliação).

A ISO/IEC 14598-6 contêm módulos que especificam o modelo de qualidade (isto é, características, subcaracterísticas e métricas externas e internas correspondentes), as informações e dados relativos à aplicação prevista do modelo e informações sobre a real aplicação do modelo.

A NBR ISO/IEC 14598 pode ser usada por organizações que produzem novos módulos de avaliação e é conveniente que seja utilizada em conjunto com a NBR ISO/IEC 9126, que descreve as características de qualidade e métricas de software, como mostra a figura 61.

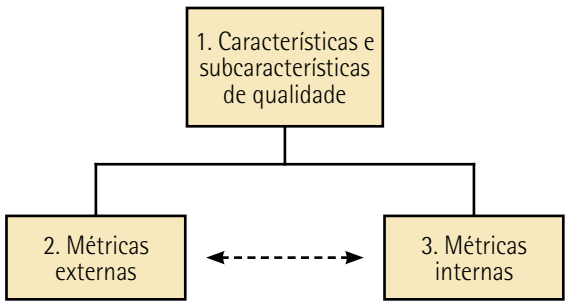

Figura 60 – NBR ISO/IEC 9126 – Características de qualidade e métricas de software

Fonte: ABNT (2001).

Para manter o sistema de software a avaliação permite validar os requisitos de qualidade e verificar se os requisitos de manutenibilidade e portabilidade estão sendo atingidos. A avaliação do software depende do estágio no ciclo de vida em que se encontra o desenvolvimento do sistema de software.

Observe a figura 61. A qualidade é avaliada de acordo com os requisitos externos e internos, que correspondem às características e subcaracterísticas da NBR ISO/IEC 9126.

A qualidade em uso deve atender as necessidades do mundo real.

A qualidade externa é avaliada apenas se o sistema estiver completo de hardware/software, ou seja, as métricas externas são aplicadas durante a execução do software.

A qualidade interna é avaliada de acordo com as ligações dos componentes (hardware, software, banco de dados e rede de computadores) do sistema de software, bem como suas respectivas métricas, por exemplo, código-fonte, desempenho do sistema, desempenho da rede, integração e outros.

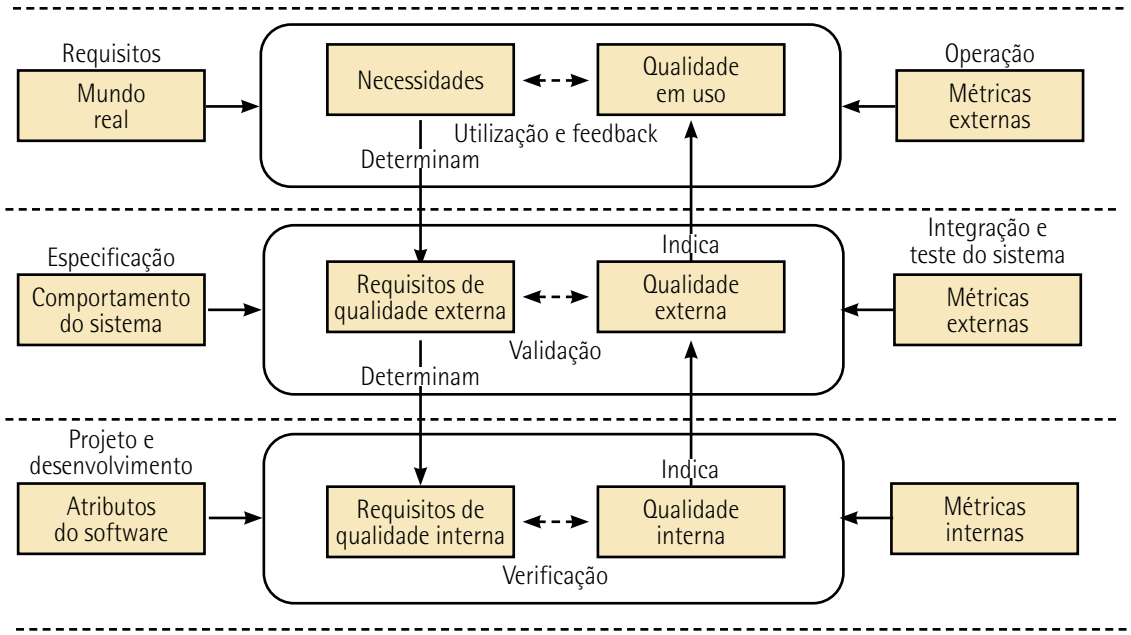

Figura 61 – Qualidade no ciclo de vida de software

Fonte: ABNT (2001).

As medidas da qualidade interna de software podem ser utilizadas como indicadores para estimar a qualidade em uso, veja a figura 62. Atingir a qualidade interna requerida contribuirá para atender aos requisitos externos do software em uso. Exemplo de medidas: modularidade, portabilidade, rastreabilidade, desempenho e outros.

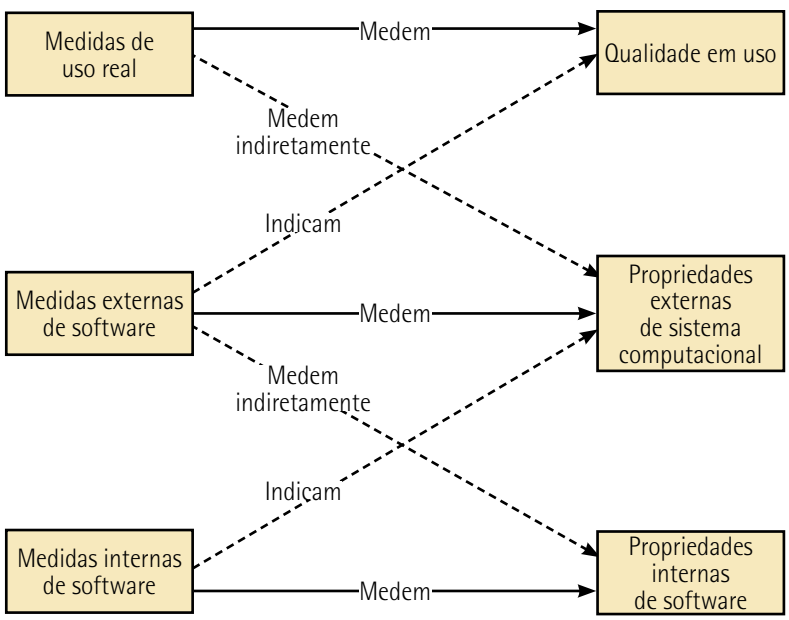

Figura 62 – Relacionamento entre medidas e atributos

Fonte: ABNT (2001).

7.2.3 NBR ISO/IEC 25000 – Guia do SQuaRE*

Esta norma diz respeito à descaracterização e medição de qualidade de produto de software. A NBR ISO/IEC 25000 é uma evolução das séries de NBR ISO/IEC 9126 e NBR ISO/IEC14598.

A reorganização da NBR ISO/IEC 25000 gera uma nova divisão de assuntos em cinco tópicos, sendo que cada divisão é composta por um conjunto de documentos e trata de um assunto em particular, como mostra o quadro a seguir:

Quadro 14 – Divisão dos documentos da NBR ISO/IEC 25000

Requisitos de qualidade	Modelo de qualidade	Avaliação
	Gerenciamento de qualidade	
	Medições	

*SQuaRE significa *Software Product Quality Requirements and Evaluation* (Requisitos de Qualidade e Avaliação de Produtos de Software).

De acordo com a Softex (2013), na apresentação do MPS.BR – Melhoria de Processo do Software Brasileiro: Guia de Aquisição, em Avaliação com as séries ABNT NBR ISO/IEC 9126 e 14598, pode-se observar como foi formulada a NBR ISO/IEC 25000, apresentando basicamente 10 conjuntos de normas, mostrados no quadro a seguir:

Quadro 15 – Normas da NBR ISO/IEC 25000

ABNT NBR ISO/IEC	DESCRIÇÃO
9126-1	Modelo de qualidade de software: referência para o processo de avaliação da qualidade de produto de software
9126-2	Métricas externas: conjunto de métricas externas que podem ser utilizadas para definição e avaliação de qualidade de produto de software
9126-3	Métricas internas: conjunto de métricas internas que podem ser utilizadas para definição e avaliação de qualidade de produto de software
9126-4	Métricas para qualidade em uso: conjunto de métricas de qualidade em uso e exemplo de processo de avaliação da qualidade em uso
14598-1	Guia de avaliação – visão geral: apresenta toda a estrutura de funcionamento da série de normas para avaliação da qualidade dos produtos de software
14598-2	Planejamento e gerenciamento de avaliações: o suporte está relacionado ao planejamento e gerenciamento de um processo de avaliação de software e a tecnologia necessária
14598-3	Processo de avaliação para desenvolvedores: uso durante o processo de desenvolvimento e manutenção de software
14598-4	Processo de avaliação para adquirentes: estabelece um processo sistemático para avaliação de produtos de software de prateleira, produtos de software sob encomenda ou, ainda, modificações em produtos já existentes
14598-5	Processo de avaliação para avaliadores: orientações para a implementação prática de avaliação de produto de software, quando diversas partes necessitam entender, aceitar e confiar em resultados de avaliação
14589-6	Documentação de módulos de avaliação: estrutura e conteúdo da documentação a ser usada na descrição dos Módulos de Avaliação

A série de normas SQuaRE estabelece um conjunto de padrões e diretrizes para a avaliação da qualidade de software e produtos relacionados. Ela foi desenvolvida para fornecer uma estrutura abrangente que permita avaliar e medir a qualidade do software de forma objetiva e consistente.

De acordo com a Softex (2013), o núcleo principal do SQuaRE é composto de quatro divisões de normas e uma sequência prevista para extensão do modelo, como mostra a figura 63:

- **ISO/IEC 2500n – Divisão Gestão da Qualidade**: engenharia de software – Requisitos e avaliação da qualidade de produto de software (SQuaRE) – Guia do SQuaRE (publicada pela ABNT).

- **ISO/IEC 2501n – Divisão Modelo de Qualidade**: define os modelos de qualidade de produto de software e qualidade de dados.

- **ISO/IEC 2502n – Divisão Medição da Qualidade**: com base nas características e subcaracterísticas, define os modelos de introdução aos elementos de medida de qualidade externa, interna e em uso, medidas de qualidade para subsidiar os usuários, medidas de qualidade de sistemas e de produto de software e medida de qualidade de dados.

- **ISO/IEC 2503n – Divisão Requisitos de Qualidade**: fornece os requisitos e recomendações para a especificação de requisitos de qualidade de produto de software.

- **ISO/IEC 2504n – Divisão Avaliação da Qualidade**: define requisitos gerais para avaliação da qualidade de produto de software, fornece requisitos e orientações para avaliação de produto de software segundo a perspectiva de desenvolvedores, adquirentes e avaliadores independentes e define a estrutura e conteúdo da documentação do módulo de avaliação.

- **ISO/IEC 2505n – ISO/IEC 25099 – Extensão do SQuaRE**: estabelece requisitos de qualidade para produto de software comercial de prateleira (COTS), casos e relatório de testes, define uma visão geral de um *framework*, utilizado para documentar a especificação da avaliação de usabilidade de sistemas interativos e define como registrar as medidas obtidas em testes de usabilidade, onde são avaliadas as características de eficácia, eficiência e satisfação num contexto de uso especificado.

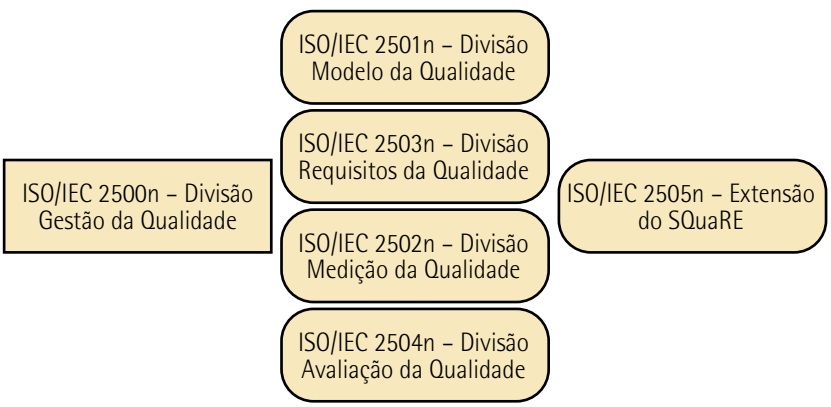

Figura 63 – Divisões e extensão do SQuaRE

8 EVOLUÇÃO, REENGENHARIA, MANUTENÇÃO E TENDÊNCIAS

A evolução, reengenharia, manutenção e tendências do software estão intrinsecamente interligadas, impulsionando a inovação e a melhoria contínua na indústria. Prometem moldar um futuro cada vez mais tecnológico e repleto de oportunidades para o desenvolvimento de soluções inovadoras.

A evolução do software é marcada por constantes melhorias e atualizações para atender às necessidades em constante mudança do mercado. A rápida inovação tecnológica e as crescentes demandas dos usuários têm levado a indústria de software a experimentar uma evolução contínua ao longo dos anos.

A reengenharia de software envolve a análise profunda e a remodelagem de softwares existentes, buscando melhorar a eficiência, qualidade e capacidade de adaptação deles às novas tendências e tecnologias. A reengenharia de software surgiu como uma abordagem estratégica para revitalizar e aprimorar sistemas legados.

A manutenção de software envolve atividades corretivas, adaptativas e preventivas, garantindo que o software continue funcionando adequadamente e acompanhe as mudanças do ambiente tecnológico. A manutenção de software também é um aspecto crucial da evolução do software.

As tendências do software oferecem oportunidades emocionantes para aprimorar a experiência do usuário, a eficiência dos sistemas e a segurança. Algumas das tecnologias que estão moldando o futuro do software se destacam em relação à inteligência artificial, à computação em nuvem, à Internet das Coisas (IoT) e à realidade virtual.

Saiba mais

Sem dúvida a Inteligência Artificial (IA) é o principal desafio dos desenvolvedores de software da atualidade. Conheça os principais objetivos e tecnologias de software que estão sendo pesquisadas, estudadas e desenvolvidas no campo da IA. Consulte:

IBM. *O que é inteligência artificial (IA)?* [s.d.]. Disponível em: http://tinyurl.com/5evry7uu. Acesso em: 18 nov. 2023.

8.1 Evolução do software

A evolução do software é importante, pois as organizações investem grandes quantias de dinheiro em seus softwares e são totalmente dependentes desses sistemas, que são ativos críticos de negócios, e fazem com que as organizações invistam nas mudanças de sistemas para manter o valor desses ativos. Consequentemente, a maioria das grandes empresas gasta mais na manutenção de sistemas existentes do que no desenvolvimento de novos sistemas (Sommerville, 2016).

Atualmente, a evolução do software é decorrente principalmente das necessidades de negócios em constante mudança e falhas devido a mudanças no ambiente operacional.

A evolução do software deve ser avaliada em conjunto com os componentes que integram o seu sistema, que também têm um certo grau de evolução. Tais componentes envolvem não só a lógica de processamento, mas também a implementação de novas funcionalidades, o hardware, o banco de dados e a rede de computadores. Consequentemente, envolvem o impacto da mudança por parte do usuário, o que pode aumentar as dificuldades operacionais, com o custo da evolução.

No ciclo de vida de evolução do software os requisitos dos sistemas de software instalados mudam à medida que o negócio e ambiente requerem, gerando, portanto, novos releases dos sistemas que incorporam as mudanças necessárias.

De acordo com Sommerville (2016), as atualizações geralmente são criadas em intervalos regulares. A engenharia de software é, portanto, um processo em espiral com requisitos, design, implementação e testes em curso durante toda a vida útil do sistema. Observe a figura 64.

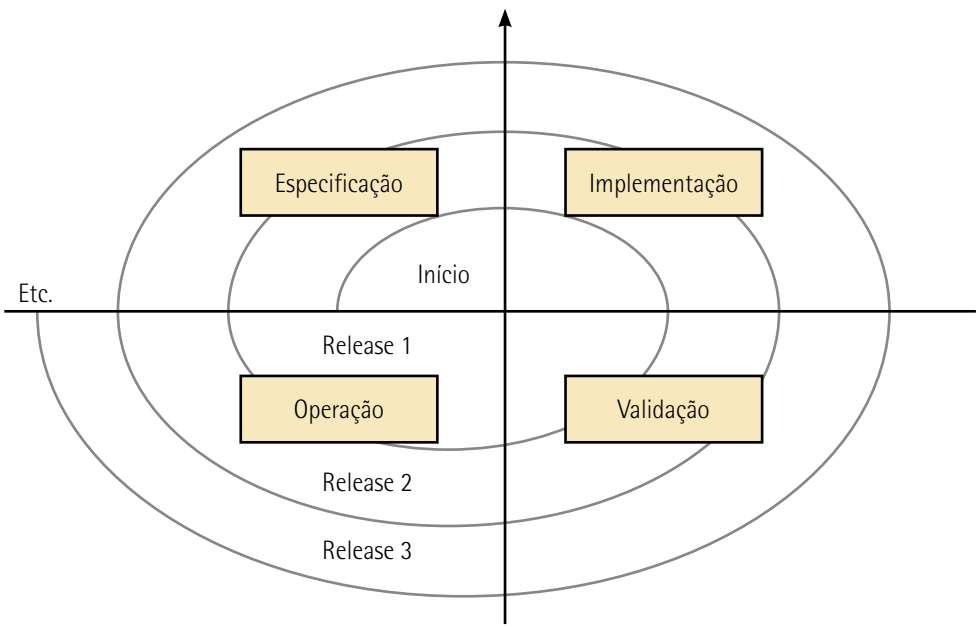

Figura 64 – Modelo em espiral de desenvolvimento e evolução

Fonte: Sommerville (2011).

As propostas de mudanças devem ser relacionadas aos componentes do sistema que necessitam ser modificados para implementar essas propostas, o que permite que o custo e o impacto da alteração sejam avaliados. Isso faz parte do processo de gerenciamento de mudanças, que também deve assegurar que as versões corretas dos componentes estejam incluídas em cada release do sistema.

8.2 Reengenharia de software

O objetivo da reengenharia de software é o de redesenhar, reestruturar e otimizar sua arquitetura, código e funcionalidades. Tem como foco modernizar o software, tornando-o mais eficiente, escalável e adaptável às demandas atuais.

Essa abordagem permite eliminar redundâncias, melhorar a qualidade e segurança do código e agregar novas tecnologias sem comprometer a funcionalidade original.

A reengenharia de software é uma poderosa ferramenta para revitalizar e prolongar a vida útil de sistemas legados, impulsionando a inovação e garantindo a competitividade das empresas em um ambiente tecnologicamente dinâmico.

Exemplo de aplicação

Vamos observar a Reengenharia dos Processos de Negócio (*Business Process Redesign* – BPR).

A Reengenharia dos Processos de Negócio consiste no alinhamento do Planejamento Estratégico de Negócios (PEN) com o Planejamento Estratégico de TI (PETI). A integração dos negócios com a tecnologia da informação deve ser acompanhada por uma infraestrutura informacional e tecnológica que dê suporte à comunicação da informação e do conhecimento. Veja a figura 65:

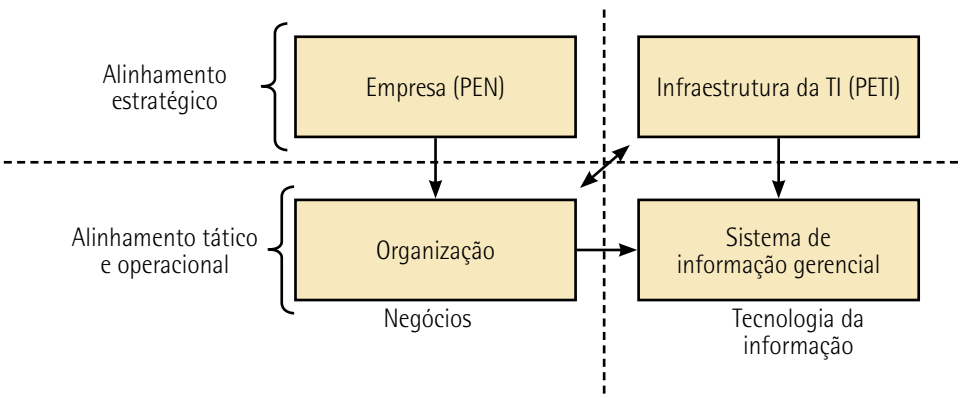

Figura 65 – Integração funcional dos negócios com a tecnologia da informação

- **Empresa**: é um conjunto de áreas de conhecimentos com objetivo de fazer negócio. A administração fornece a estratégia de negócio. Os administradores percebem os desafios e necessidades do mercado, estabelecem a estratégia organizacional, com o uso dos recursos materiais, humanos, tecnológicos e financeiros, com objetivos de operacionalizar a empresa para atender o desafio ou necessidade do mercado.

- **Organização**: são formadas pelos processos organizacionais. O processo organizacional seleciona as atividades ou operações necessárias de uma determinada área de conhecimento, bem como sua capacidade para atender a administração. Por meio da organização são elaboradas as atividades e tarefas da empresa.

- **Infraestrutura da Tecnologia da Informação ou Estratégia de TI**: é o que o gerente utiliza para enfrentar as mudanças. A tecnologia da informação fornece a infraestrutura tecnológica do sistema de informação para converter as atividades da empresa em funções do sistema de informação.

- **Sistema de Informação Gerencial**: é o resultado das regras de negócios alinhadas e automatizadas pela tecnologia da informação.

Um modelo de processo da reengenharia de software com suas principais atividades é proposto por Pressman (2011) e apresentado na figura 66. O processo tem início na Análise do Inventário, que, por meio do histórico de mudanças no software, investimentos e critérios de manutenibilidade busca alocar recursos para a reengenharia do software.

Um destaque nesse modelo é para a engenharia reversa, que por diversas vezes é aplicada quando se parte da informação ou do conhecimento para os dados que a podem gerar. Essa atividade é também amplamente utilizada no processo de manutenção do software. Em termos industriais, a "engenharia reversa tem suas origens no mundo do hardware, para conhecer os segredos de projeto e fabricação do concorrente" (Pressman, 2011).

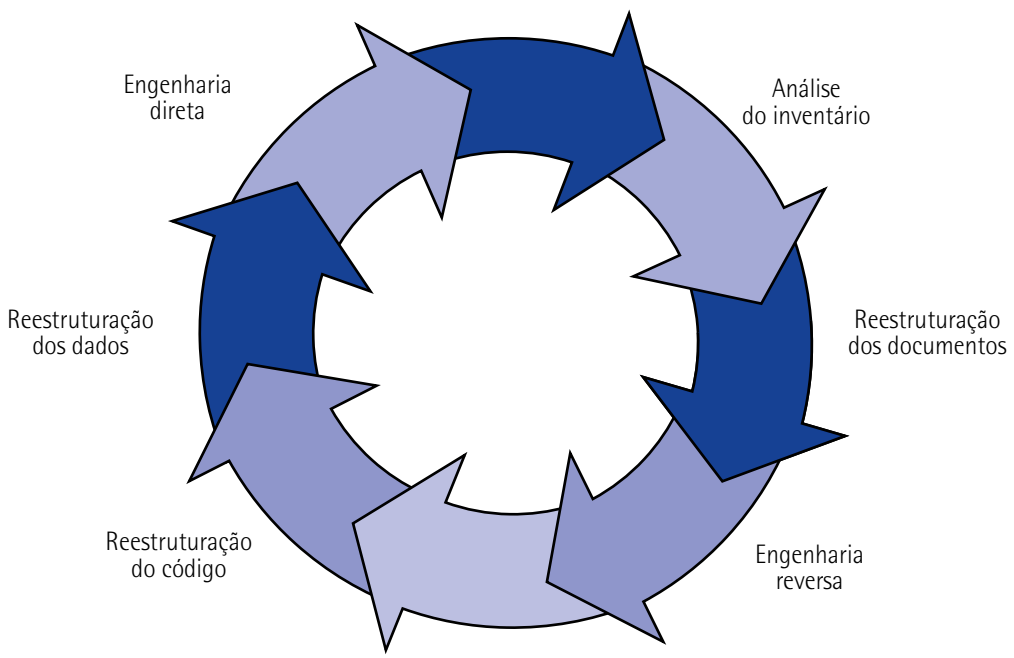

Figura 66 – Modelo de processo de reengenharia de software e principais atividades

Fonte: Pressman (2011).

Observação

Uma abordagem mais detalhada sobre a reestruturação do software pode ser lida no tópico 1.2.1 Dualidade do software com o hardware.

A engenharia direta é um processo de melhoria contínua do software, em que as mudanças ocorrem por meio de iterações que envolvem: verificação de informações do projeto, implementação ou manutenção de funcionalidades, melhoria do desempenho e da usabilidade.

Nos primórdios da era do computador, os sistemas baseados em computador eram desenvolvidos pela administração orientada ao hardware, hoje em dia vemos um índice de obsolescência muito alto do computador comparado com demais produtos do mercado, devido à dualidade do software com o hardware. "Se fizer mudanças no software ou no hardware, é provável que tenha que atualizar algum dos dois elementos".

O fato é que há demanda por novas funcionalidades de software e implementações de código (adaptações, manutenção ou atualizações). Com o tempo, esse aumento de mudanças para manter o sistema de software durante seu ciclo de vida é impulsionado pelo próprio software, que passa a exigir melhorias ou até mesmo a troca do hardware.

Resumidamente, a distribuição dos custos para o desenvolvimento baseados em computador mudou drasticamente para o software e, nesse contexto, para a reestruturação do sistema de software considera-se o gerenciamento de sistemas legados.

Sistemas legados são sistemas antigos que ainda operam e precisam sempre estar ativos e com bom desempenho. A reengenharia do software se faz útil nesse caso, porém alguns parâmetros devem ser avaliados antes da prática da reengenharia ou manutenção do software.

Os sistemas legados não são apenas sistemas de software, mas são sistemas sociotécnicos mais amplos, que incluem hardware, software, bibliotecas ou outro software de suporte aos negócios (Sommerville, 2016).

Na reengenharia as organizações têm que decidir como obter o melhor retorno sobre o seu investimento. Isto implica fazer uma avaliação realista dos seus sistemas legados para decidir sobre a melhor estratégia para a evolução destes sistemas. De acordo com Sommerville (2016), existem quatro estratégias:

1. Descartar completamente o sistema: aplicado quando o sistema não contribui de forma eficaz nos processos empresariais. Exemplo: computadores antigos que operavam um sistema legado e, devido a atualizações de software, esse sistema legado passou a não ser mais útil.

2. Deixar o sistema inalterado e continuar com a manutenção regular: aplicado quando o sistema legado ainda é necessário, bastante estável e os usuários solicitam poucas mudanças.

3. Reestruturar o sistema para melhorar sua manutenibilidade: aplicado quando a qualidade do sistema legado estiver deteriorada por diversas mudanças. Situação observada principalmente quando os usuários passam a reclamar do sistema de software devido ao aumento de falhas, perda de dados e queda no desempenho.

4. Substituir todo ou parte do sistema por um novo: aplicado quando fatores como novo hardware, novas bibliotecas ou novas funcionalidades implementadas no sistema legado impedem o funcionamento do sistema por completo. Neste caso, a análise do custo/benefício com base nos itens 1, 2 e 3 deve ser reavaliada e considerar como opções pesquisas no mercado de aplicações de software completos e comercializados.

8.3 Manutenção de software

Das diversas peculiaridades da engenharia de software que diferem das demais engenharias, está o processo de manutenção do software. É de se esperar que ao comprar um produto, ele venha sem qualquer problema e pronto para usar. Pois é, com o produto software não é assim!

Na implantação de um sistema ou até na simples instalação de um novo software, a manutenção começa quase em seguida, e abrange problemas de configurações, adaptações ao ambiente operacional, interfaces não operacionais e o maior deles, que são novos requisitos de mudanças por parte do usuário, seja porque gostou do software e quer explorar sua capacidade e usabilidade, ou os que não gostaram e solicitam adaptações, implementação de novas funções, novas interfaces e outros.

A manutenção de software é o processo geral de mudança em um sistema depois que ele é liberado para uso. O termo geralmente se aplica ao software customizado em que grupos de desenvolvimento separados estão envolvidos antes e depois da liberação (Pressman, 2016).

A manutenção pode ser corretiva, preventiva ou adaptativa, permitindo ajustes conforme as mudanças tecnológicas e demandas dos usuários. Por meio da manutenção de software, as organizações asseguram a qualidade, segurança e usabilidade de suas aplicações, promovendo um ambiente de TI confiável e eficiente.

Na manutenção corretiva o objetivo é restaurar o software ao seu estado funcional normal e garantir que ele atenda às expectativas dos usuários, o que corresponde à atividade de depuração de falhas (debug) do software, que inclui:

- **Rastreamento do erro (ou bug)**: identificação da causa do problema.
- **Diagnóstico das causas do erro**: avaliação da causa do problema, sua gravidade, prioridade, análise do risco e impactos no comportamento do software.
- **Correção**: implementação e testes unitários e posteriormente testes de integração.

- **Reflexão**: aplicação de medidas corretivas para solucionar o problema e análise por conta dos desenvolvedores e cliente para validar a mudança no software.

Exemplo: clientes e usuários passaram a ter problemas no fechamento de seus carrinhos de compras em uma aplicação de e-commerce. O erro ocorria na transação de validação do cartão de crédito, mesmo estando em condições regulares, o que nunca tinha acontecido!

A equipe de manutenção foi acionada para corrigir o bug e restaurar a funcionalidade da aplicação.

A manutenção preventiva inclui atividades como análises de desempenho, revisões de código, atualizações de segurança e outras medidas que visam manter o software eficaz e com bom desempenho. Envolve ações proativas para evitar possíveis problemas futuros no software.

Inclui-se aqui a manutenção preventiva por refatoração, uma prática que envolve a reestruturação e melhoria do código-fonte do software sem alterar seu comportamento externo. É o processo de fazer ajustes no código para torná-lo mais legível, eficiente, organizado e sustentável, sem afetar a funcionalidade do software.

Exemplo: por se tratar de sistemas grandes e complexos, a computação em nuvem (*cloud computing*) tem equipes de manutenção que realizam verificações regulares no código-fonte para identificar áreas que possam apresentar vulnerabilidades de segurança, corrigindo-as antes que se tornem um problema.

A manutenção adaptativa faz alterações necessárias no software para mantê-lo operacional devido a mudanças no ambiente operacional, como atualizações de sistemas operacionais, inserção de novas bibliotecas, instalações de novos programas diferentes até do software em manutenção, mudanças de requisitos legais ou tecnológicas.

Exemplo de aplicação

Manutenção da automatização em supermercados: nos caixas de mercado, os PDVs (ponto de vendas), leitores de código de barras espalhados para consulta de preços dos produtos por parte dos clientes e sistemas administrativos estão ligados ao mesmo banco de dados. Contudo, em uma nova aliança de empresas no ramo de mercados foi necessário compartilhar dados com outros ambientes operacionais em um sistema distribuído.

Neste caso, a equipe de manutenção foi acionada para adaptar novos requisitos de sistema ao software.

Em uma manutenção corretiva, além do uso da ferramenta de debug, na análise de defeitos, é sugerido também o uso da técnica de análise com o diagrama de causa e efeito orientado pelo PMBOK (2017), como mostra a figura 67, cujo resultado é: "a qualidade do produto não corresponde aos requisitos".

O diagrama apresentado para a análise da qualidade de um produto genérico pode ser adaptado à análise de defeitos do produto software, por incluir muitas variáveis, ligações e um certo grau de

complexidade. O diagrama causa e efeito permite detalhar e especificar a causa do defeito para uma possível manutenção.

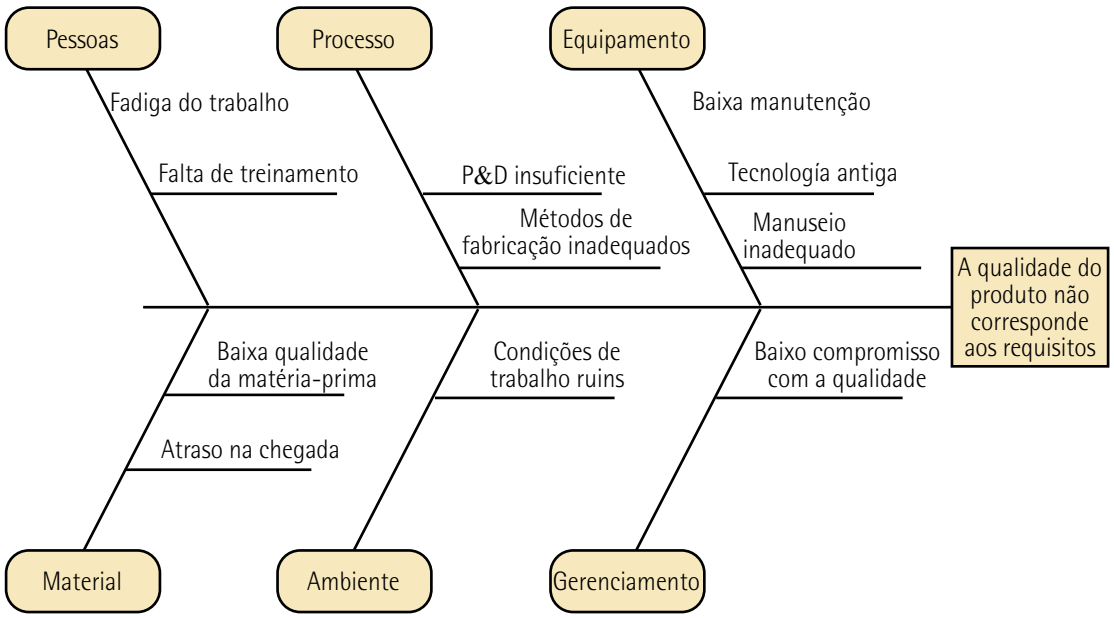

Figura 67 – Técnica de análise da qualidade do produto com o diagrama de causa e efeito (chamado também de diagrama de Ishikawa ou diagrama de espinha de peixe)

Fonte: PMBOK (2017).

Em relação aos tipos de manutenção que ocorrem durante todo o ciclo de vida operacional do software, Pressman (2011) e Sommerville (2016) estão em comum acordo de que os tipos de manutenção do software são motivados por três tipos fundamentais, mostrados no quadro a seguir.

Quadro 16 – Tipos de manutenção de software

Tipo de manutenção	Descrição
Manutenção para reparar os defeitos no software	A correção de erros de codificação é um processo relativamente barato comparado com os erros de projeto Os maiores custos estão nos erros de requisitos, pois implicarão um novo projeto
Manutenção para adaptar o software a um ambiente operacional diferente	É a típica manutenção de adaptação sofrida por alguma alteração no software de apoio, tal como o sistema operacional, banco de dados ou mesmo o próprio hardware
Manutenção para fazer acréscimos à funcionalidade do sistema ou modificá-la	Na alteração dos requisitos, devido a mudanças organizacionais, ou nos negócios, que são bastante constantes, ocorre a manutenção mais comum entre todas as outras

Adaptado de: Pressman (2011) e Sommmerville (2016).

8.4 Tendências do software

Atualmente observamos muitas pesquisas em torno do software, algumas delas são demoradas, complexas e não atingem seus objetivos.

Quando é introduzida uma tecnologia bem-sucedida, o conceito inicial transforma-se em "ciclo de inovação", sendo razoavelmente previsível (Gaines, 1995 *apud* Pressman, 2011).

Devido aos avanços tecnológicos e às mudanças nas demandas dos usuários e do mercado, as tendências do software na engenharia e em várias outras áreas estão em constante evolução. Algumas tendências notáveis incluem:

- **Inteligência artificial e aprendizado de máquina**: a inteligência artificial – IA (do inglês *Artificial Inteligence* – AI) "faz uso de algoritmos não numéricos para resolver problemas complexos que não sejam favoráveis à computação ou à análise direta" (Pressman, 2011). O aprendizado de máquina (do inglês *machine learning* – ML) é uma ramificação da IA, que tem o foco no desenvolvimento de algoritmos e técnicas que permitem aos computadores aprender com grandes volumes de dados, analisar para identificar padrões e tomar decisões ou fazer previsões com base nesses padrões.

A IA e o ML estão transformando a maneira como o software é desenvolvido e usado. Algoritmos de IA são incorporados em aplicativos para melhorar a automação, análise de dados e interação com os usuários.

Exemplo: na IA a área mais ativa é a dos Sistemas Especialistas, também chamados de sistemas baseados em conhecimento. São sistemas que resolvem problemas, capturando o conhecimento para um domínio muito específico e limitado da perícia.

- **Computação em nuvem (*Cloud Computing*)**: os serviços oferecidos reúnem diversos recursos de processamento, aplicações e armazenamento em servidores, que podem ser acessados de qualquer lugar do mundo pela internet. Além de permitir que aplicativos e serviços sejam entregues pela internet, a computação em nuvem proporciona escalabilidade, flexibilidade e acessibilidade a partir de dispositivos variados.

A computação em nuvem tem sido adotada por diversas empresas, e alguns desses sistemas de software são disponibilizados ao público para uso geral. Daqueles que mais se destacam no Brasil estão o Google Drive e o Outlook Drive da Microsoft, como mostra a figura 68.

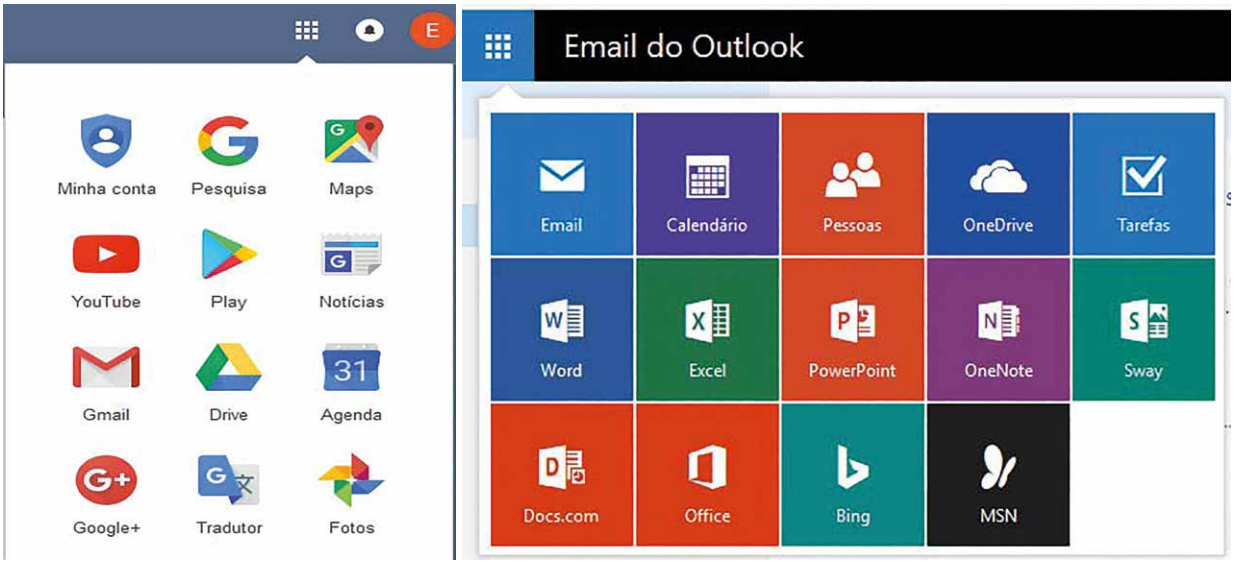

Figura 68 – Telas de sistema de software para computação em nuvem (cloud computing) de uso público: Google Drive (figura da esquerda) e Outlook Drive da Microsoft (figura da direita)

- **Aplicações móveis**: o uso crescente de dispositivos móveis impulsionou a criação de aplicativos móveis eficientes, intuitivos e altamente funcionais, abrangendo áreas como e-commerce, saúde, redes sociais e entretenimento.

- **Internet das Coisas (IoT - *Internet of Things*)**: o controle e a interconexão pela internet de dispositivos inteligentes está impulsionando a demanda por software, de forma a coletar, analisar e atuar sobre dados gerados por esses dispositivos em setores como casas inteligentes, manufatura e cidades inteligentes.

- **Desenvolvimento Ágil e DevOps**: a abordagem ágil permite um desenvolvimento mais iterativo e colaborativo, enquanto o DevOps integra desenvolvimento e operações para acelerar a entrega e melhorar a qualidade do software.

- **Interfaces de Usuário Avançadas**: interfaces de usuário como: realidade virtual (do inglês *virtual reality* – VR) e realidade aumentada - RA (do inglês *augmented reality* – AR) e interfaces de voz estão se tornando mais comuns, proporcionando interações mais imersivas e naturais. Exemplo: como mostra a figura 69, a realidade aumentada (RA) permite integrar informações digitais em realidade virtual ao ambiente real. Com isso é possível analisar uma peça de motor virtual em um motor real, ampliar órgãos humanos, de animais, plantas e objetos em geral.

Figura 69 – Análise detalhada de um objeto ampliado com o recurso da realidade aumentada

Disponível em: http://tinyurl.com/32jucpns. Acesso em: 4 jan. 2023.

- **Segurança cibernética**: com a crescente preocupação com a segurança de dados, o desenvolvimento de software seguro é uma prioridade. Incluindo técnicas avançadas de criptografia, autenticação e proteção contra ameaças, a segurança cibernética é um campo crítico em evolução na proteção de sistemas computacionais, com objetivo de salvaguardar a integridade, confidencialidade e disponibilidade das informações digitais, garantindo que as operações online sejam seguras e confiáveis.

- **Blockchain**: principalmente no setor financeiro, o blockchain está sendo utilizado para garantir a segurança e a rastreabilidade de transações e processos. Blockchain é uma tecnologia revolucionária que oferece uma abordagem descentralizada e segura para o registro e verificação de transações e informações.

- **Sustentabilidade e ética**: as preocupações ambientais e éticas estão moldando a forma como o software é desenvolvido, com uma ênfase crescente em práticas de programação responsável e sustentável.

- **Análise de dados e inteligência de negócios**: a capacidade de coletar e analisar dados em tempo real está ajudando as empresas a tomarem decisões informadas e preverem tendências futuras.

Exemplo: o BI (*Business Intelligence*) é um exemplo avançado de sistema integrado de gestão, tendo em vista que é composto de várias ferramentas que combinam diversos tipos de dados para análise, comparação e simulação de resultados que levam diretamente à tomada de decisão. O BI proporciona ao gestor ferramentas para extrair dados de forma ágil e confiável, fornecendo uma visão global do andamento dos negócios da empresa.

Essas tendências do software refletem a crescente interconexão entre tecnologia e sociedade, moldando a maneira como desenvolvemos, entregamos e usamos software em diversos setores e aplicações.

Lembrete

A reengenharia de software envolve a análise profunda e a remodelagem de softwares existentes, buscando melhorar a eficiência, qualidade e capacidade de adaptação deles às novas tendências e tecnologias.

Resumo

A interação entre seres humanos e computadores tem evoluído significativamente nas últimas décadas, transformando a maneira como utilizamos a tecnologia e influenciando diretamente a qualidade das soluções de software. A qualidade e a avaliação desses sistemas são apresentadas como fatores determinantes para o sucesso de um produto no mercado. A unidade III aborda como principais tópicos a Interface Homem-Computador (IHC) e a evolução, reengenharia, manutenção e tendências do software.

A IHC desempenha um papel crucial ao garantir que os sistemas sejam acessíveis, usáveis e eficazes para os usuários. Esse tópico aborda aspectos como princípios da interação humano-computador, qualidade em uso, modos de acesso, retorno de investimento e normas de qualidade. A IHC concentra-se na criação de sistemas que proporcionem uma interação intuitiva e eficiente entre humanos e computadores. Os princípios da IHC abrangem a usabilidade, a acessibilidade e a comunicabilidade, visando garantir que os sistemas atendam às necessidades e às expectativas dos usuários. Além disso, os modos de acesso dos sistemas operacionais desempenham um papel central na maneira como os usuários interagem com os dispositivos, incluindo interfaces gráficas, comandos de voz, gestos e outros métodos. A qualidade em uso de um sistema vai além de sua funcionalidade técnica, considerando aspectos que influenciam diretamente a experiência do usuário. A usabilidade se refere à facilidade com que os usuários podem interagir e realizar tarefas no sistema. A comunicabilidade envolve a clareza das informações transmitidas pela interface. A acessibilidade garante que o sistema possa ser utilizado por pessoas com diferentes capacidades e necessidades. Além disso, o Retorno de Investimento (ROI) avalia os benefícios financeiros resultantes de investimentos em melhoria de qualidade e usabilidade. As normas internacionais desempenham um papel importante na definição de critérios de qualidade e avaliação de sistemas de software. A ISO 9126 estabelece um modelo de qualidade que abrange características como funcionalidade, confiabilidade, usabilidade e eficiência. A ISO 14598 define diretrizes para avaliação de produtos de software, considerando aspectos como planejamento, execução e documentação do processo de avaliação. A ISO 25000, também conhecida como SquaRE, é uma série de normas que integram a ISO 9126 e a ISO 14598, fornecendo um modelo mais abrangente para avaliação de qualidade de software.

A evolução dos sistemas de software é uma realidade constante. A reengenharia é a abordagem de revitalização de sistemas legados, tornando-os mais eficientes e atualizados. A manutenção garante que os sistemas continuem funcionando corretamente e atendendo às necessidades dos usuários. As tendências atuais incluem a adaptação a dispositivos móveis, a integração de inteligência artificial e a incorporação de interfaces de realidade virtual e aumentada.

Em síntese, a qualidade da interação humano-computador e a avaliação rigorosa dos sistemas de software são essenciais para o sucesso no mercado atual. As normas de qualidade estabelecem critérios objetivos de avaliação, enquanto a evolução, reengenharia e manutenção garantem a relevância contínua dos sistemas. Com a evolução da tecnologia, espera-se que a melhoria da qualidade em uso e a atenção à experiência do usuário permaneçam como prioridades, moldando o futuro do desenvolvimento de software.

Exercícios

Questão 1. (Idecan/2023, adaptada) Rafael trabalha como implementador de software e recebe a demanda de formar parte do time de qualidade de software de um projeto. O roteiro de trabalho de Rafael pede que ele foque seu trabalho no atributo de qualidade de software que abrange a capacidade do produto de ser compreendido, aprendido e operado e ter interface atraente ao usuário.

Selecione a alternativa que mostra o atributo de qualidade de software citado no roteiro de trabalho de Rafael.

A) Usabilidade.

B) Portabilidade.

C) Segurança.

D) Resiliência.

E) Proteção.

Resposta correta: alternativa A.

Análise da questão

A usabilidade é um atributo de qualidade de software que pode ser definido como a capacidade de um produto de software ser facilmente compreendido, aprendido e operado pelo usuário, além de se mostrar um produto atraente e amigável. Esse atributo visa a proporcionar uma experiência eficaz e satisfatória ao usuário. Portanto, a usabilidade é o atributo de qualidade de software citado no roteiro de trabalho de Rafael.

Questão 2. (FAURGS/2018, adaptada) No contexto da manutenção de software, avalie as afirmativas.

I – A manutenção de software é o processo geral de mudança em um sistema depois que ele já foi liberado para uso.

II – A manutenção de software ocupa uma proporção menor dos orçamentos de TI do que o desenvolvimento e, portanto, os esforços durante o desenvolvimento do sistema para a produção de um sistema manutenível não reduzem os custos gerais durante a vida útil do sistema.

III – Existem três diferentes tipos de manutenção de software: correção de defeitos, adaptação ambiental e adição de funcionalidade.

É correto o que se afirma em:

A) I apenas.

B) III apenas.

C) I e III apenas.

D) II e III apenas.

E) I, II e III.

Resposta correta: alternativa C.

Análise das afirmativas

I – Afirmativa correta.

Justificativa: a manutenção de software envolve a modificação e a atualização de um sistema de software após ele já ter sido liberado para seus usuários, o que inclui correções de defeitos, adaptação ambiental e adição de novas funcionalidades.

II – Afirmativa incorreta.

Justificativa: a manutenção de software pode ocupar uma porção significativa dos orçamentos de TI. A produção de um sistema manutenível durante o desenvolvimento pode reduzir os custos durante a vida útil do sistema. Investir em boas práticas de desenvolvimento e de design pode levar a um menor número de correções e de modificações ao longo do processo de manutenção.

III – Afirmativa correta.

Justificativa: podemos classificar a manutenção de software em três tipos, elencados a seguir.

- **Manutenção para reparar defeitos**: é a manutenção voltada a corrigir erros de codificação e de requisitos de software.

- **Manutenção para adaptar o software a um ambiente diferente**: é a manutenção necessária em razão de alguma alteração no sistema operacional, no banco de dados ou no hardware sobre o qual o software opera.

- **Manutenção para fazer acréscimos à funcionalidade do sistema**: é a manutenção voltada a atender à modificações de requisitos, devido a mudanças organizacionais.

REFERÊNCIAS

Textuais

ABNT. *NBR ISO/IEC 9126-1*: Engenharia de software – Qualidade de produto. Rio de Janeiro: ABNT, 2003.

ABNT. *NBR ISO/IEC 9241-11*: Requisitos Ergonômicos para Trabalho de Escritórios com Computadores. Rio de Janeiro: ABNT, 2002.

ABNT. *NBR ISO/IEC 12207*: Tecnologia de informação – Processos de ciclo de vida de software. Rio de Janeiro: ABNT, 1998.

ABNT. *NBR ISO/IEC 14598-1*: Tecnologia de informação – Avaliação de produto de software. Rio de Janeiro: ABNT, 2001.

ACM SIGCHI. *ACM SIGCHI Conference 1992*. [s.d.]. Disponível em: https://sigchi.org/. Acesso em: 2 fev. 2018.

BAHN, C. *830-1998 - Prática recomendada do IEEE para especificações de requisitos de software*. IEEE SA, 9 dez. 2009. Disponível em: http://tinyurl.com/2xcfb8ey. Acesso em: 29 jun. 2020.

BRASIL. *Software Público Brasileiro*. Brasília, 2011. Disponível em: http://tinyurl.com/3ray8bmn. Acesso em: 17 nov. 2023.

BECK, K. *Manifesto para Desenvolvimento Ágil de Software*. [s.d.]. Disponível em: http://tinyurl.com/5h2wmap3. Acesso em: 30 jun. 2020.

BEZERRA, E. *Princípios de Análise e Projetos de Sistemas com UML*. Rio de Janeiro: Campus-Elsevier, 2003.

BOOCH, G. *UML – Guia do Usuário*. São Paulo: Campus, 2002.

CAMARGO, R. *Extreme Programming: quais principais regras e valores?* 3 out. 2019. Disponível em: http://tinyurl.com/y8rdz8mc. Acesso em: 30 jun. 2020.

CARVALHO, C. *Interface Homem Computador – IHC: Interação*. [s.d.]. Disponível em: http://tinyurl.com/4uahuafb. Acesso em: 21 jan. 2021.

CMDPD. *Acessibilidade direito de todos!* 5 ago. 2014. Disponível em: http://tinyurl.com/bd7dfr7m. Acesso em: 21 jan. 2021.

CMMI. *CMMI for Systems Engineering/Software Engineering, Version 1.02. Staged Representation*. Pittsburgh: Pittsburgh: Carnegie Mellon Software Enginnering Institute, 2000.

CMMI. *CMMI para Desenvolvimento – Versão 1.2 – Melhoria de processos visando melhores produtos*. Pittsburgh: Carnegie Mellon Software Enginnering Institute, 2006.

CÔRTES, M. L. *Modelos de Qualidade de Software:* ISO 15504. São Paulo: IC-UNICAMP, 2017.

COSTA, O. W. D. *JAD – Joint Application Design*. Rio de Janeiro: IBPI Press, 1994.

DELUCA, J. A importância da engenharia de software: *FDD Feature-Driven Development*. [s.d.]. Disponível em: http://tinyurl.com/2csmzwvz, 7 out. 2008. Acesso em: 30 jun. 2020.

FARAHNEH, H. O.; ISSA, A. A. A Linear Use Case Based Software Cost Estimation Model. *World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering*, [s.l.], v. 5, n. 1, 2011.

FIORINI, S. *et al. Engenharia de Software com CMM*. Rio de Janeiro: Brasport, 1998.

FORTUNATO, C. *Framework DECIDE e Avaliação*. 2020. Disponível em: http://tinyurl.com/njxheac. Acesso em: 26 jan. 2021.

FOURNIER, R. *Desenvolvimento e manutenção de sistemas estruturados*. São Paulo: Makron Books, 1994.

HOUAISS, A. *Pequeno dicionário Houaiss da Língua Portuguesa*. São Paulo: Moderna, 2015.

HUMPHREY, W. S. *A Discipline for Software Engineering*. Boston: Addison-Wesley Professional, 1995.

IBM. *Processo iterativo controlado para desenvolvimento de software*. [s.d.]. Disponível em: http://tinyurl.com/5b4faez8. Acesso em: 8 jan. 2024.

IBM. *Rational Software Architect Standard Edition*. [s.d.]. Disponível em: http://tinyurl.com/yck4jbr5. Acesso em: 11 nov. 2023.

JOHNSON, P. Seis características presentes em todos os softwares bem escritos. *Computerworld*, 25 set. 2015. Disponível em: http://tinyurl.com/mzjb3v2y. Acesso em: 12 abr. 2020.

KOSCIANSKI, A. *Qualidade de Software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. 2. ed. São Paulo: Novatec Editora, 2007.

KOSCIANSKI, A. *et al. Guia para utilização das normas sobre avaliação de qualidade de produto de software – NBR ISO/IEC 9126 e NBR ISO/IEC 14598*. Curitiba: Associação Brasileira de Normas Técnicas, 1999.

KRUCHTEN, P. *Introdução ao RUP: Rational Unified Process*. 2. ed. Rio de Janeiro: Ciência Moderna, 2001.

KRUCHTEN, P. *The Rational Unified Process: An Introduction*. 2. ed. Boston: Addison-Wesley Professional, 2000.

KRUG, S. *Não me faça pensar*. São Paulo: Alta Books, 2006.

LARMAN, C. *Utilizando UML e padrões*: uma introdução à análise e ao projeto orientados a objetos e ao processo unificado. 2. ed. Porto Alegre: Bookman, 2007.

MICROSOFT. *HoloLens 2.* [s.d.]. Disponível em: http://tinyurl.com/yv3jpf4p. Acesso em: 18 nov. 2023.

NEVES, D. L. F. *Interação humano-computador. Usabilidade, Comunicabilidade e Acessibilidade*. [s.d.]. Disponível em: http://tinyurl.com/243fx56d. Acesso em: 2 fev. 2018.

OMG. *Categoria de modelagem de negócios: especificações associadas*. [s.d.]. Disponível em: http://tinyurl.com/4tf5vmre. Acesso em: 29 jun. 2020.

PAGANI, T. *Qualidade em uso: expandindo a usabilidade*. [S.D.]. DISPONÍVEL em: http://tinyurl.com/y5ssysx3. Acesso em: 2 fev. 2018.

PAULA FILHO, W. de P. *Engenharia de software:* fundamentos, métodos e padrões. 3. ed. Rio de Janeiro: LTC, 2015.

PAULK, M. C. *et al. The Capability Maturity Model for Software v1.1*. Pittsburgh: Software Engineering Institute Customer Relations, 1993.

PFLEEGER, S. L. *Engenharia de software:* teoria e prática. 2. ed. São Paulo: Pearson, 2004.

PIEAS. *Software Development Methodologies, Computer Science*. [s.d.]. Disponível em: https://shre.ink/r7OY. Acesso em: 5 nov. 2017.

PINTO, M. *Introdução ao debugging de software*. [s.d.]. Disponível em: http://tinyurl.com/4ery82pd. Acesso em: 2 set. 2016.

PMBOK. *Project Management Body of Knowledge*. Um guia do conhecimento em gerenciamento de projetos. 4. ed. Atlanta: Project Management Institute, 2010.

PMBOK. *Um guia do conhecimento em gerenciamento de projetos – Guia PMBOK*. 6. ed. Atlanta: Project Management Institute, 2017.

PRESSMAN, R. S. *Engenharia de software*. 5. ed. Rio de Janeiro: McGraw-Hill, 2002.

PRESSMAN, R. S. *Engenharia de software*. 6. ed. Rio de Janeiro: McGraw-Hill, 2007.

PRESSMAN, R. S. *Engenharia de software:* uma abordagem profissional. 7. ed. Rio de Janeiro: McGraw-Hill, 2011.

REZENDE, D. A. *Engenharia de software e sistemas de informação*. 3. ed. Rio de Janeiro: Brasport, 2005.

ROCHA, R. S. Modelagem do processo de desenvolvimento e manutenção de software para a UINFOR/UESB. *Cadernos NPGA*, Bahia, v. 3, n. 2, 2006.

SILVA, T. A. S. *Modelos UML estáticos vs. dinâmicos*. [s.d.]. Disponível em: http://tinyurl.com/yck6bwpe. Acesso em: 23 jun. 2020.

SLACK, N. *et al. Administração da produção*. São Paulo: Atlas, 2006.

SOFTEX. *MPS.BR – Melhoria de Processo do Software Brasileiro: Guia de Aquisição*. [s.d.]. Disponível em: http://tinyurl.com/33av6ztw. Acesso em: 9 jan. 2023.

SOMMERVILLE, I. *Engenharia de software*. 9. ed. São Paulo: Pearson, 2011.

SOMMERVILLE, I. *Software engineering*. 10. ed. New Jersey: Pearson, 2016.

SOUZA, C. S. *et al. Projeto de interfaces de usuário:* perspectivas cognitivas e semióticas. Rio de Janeiro: PUC, 2000.

STAIR, R. M.; REYNOLDS, G. W. *Princípios de Sistemas de Informação:* uma abordagem gerencial. 3. ed. São Paulo: LTC, 2006.

STOJANOVIC, S. *Versão ISO 9001:2015:* lista de materiais úteis. [s.d.]. Disponível em: http://tinyurl.com/yc2aj34n. Acesso em: 25 jul. 2023.

TANENBAUM, A. S. *Organização estruturada de computadores*. São Paulo: Pearson, 2013.

TCU. *Manual de Medição Funcional de Software. Versão 4.0*. Distrito Federal: Tribunal de Contas da União, 2016.

TIMP, A. *uTip - Early Function Point Analysis and Consistent Cost Estimating*. IFPUG, 2015. Disponível em: http://tinyurl.com/rp2y5nw. Acesso em: 15 jul. 2023.

TOTVS. *Plano de Gerenciamento de Riscos: como elaborar?* Totvs, 2023. Disponível em: http://tinyurl.com/mk5h4e8y. Acesso em: 17 nov. 2023.

TRTPR. *Análise de Pontos de Função (APF)*. [s.d.]. Disponível em: http://tinyurl.com/2b6s6wzz. Acesso em: 15 jul. 2023.

VARGAS, R. *Manual prático do plano de projeto:* utilizando o PMBOK guide. 4. ed. Rio de Janeiro: Brasport, 2011.

VAZQUEZ, C. E.; SIMÕES, G. S.; ALBERT, R. M. *Análise de pontos de função*. São Paulo: Érica, 2013.

VERSOLATTO, F. R. *Projeto de sistemas orientado a objetos*. São Paulo: Sol, 2015.

Informações:
www.sepi.unip.br ou 0800 010 9000