UNIDADE I

Engenharia de Software

Prof. Me. Edson Moreno

Introdução

- Em 1947, ao término da Segunda Grande Guerra, os computadores se tornaram públicos e a partir daí o mundo testemunhou uma crescente dependência e adoção do software em diversas áreas do conhecimento.
- A demanda por softwares cresceu e ainda cresce nos tempos atuais. A humanidade prosperou nesses últimos 80 anos o que não o fez em 5.000 anos. O software foi o responsável por essa transformação positiva.
- Em resposta à crescente demanda por softwares e acompanhando toda essa evolução, a Engenharia de Software proporcionou uma explosão de ferramentas e técnicas que revolucionaram a forma de desenvolver softwares.

- A proposta da disciplina é capacitar o aluno no conhecimento e práticas profissionais da Engenharia de Software.
- O slide 1 corresponde à Unidade I em resumo dos capítulos: 1. Fundamento da Engenharia de Software; 2. Ciclo de Vida do Software; 3. Organização e o Processo de Desenvolver Software; e 4. Customização do Software.

1. Fundamentos da engenharia de software

Em Fundamentos da engenharia de software serão apresentados:

- O conceito de produto software e a dualidade entre as engenharias de software e de sistemas;
- Os processos de desenvolvimento e sua forma de produção, bem como suas características, aplicações, suporte e manutenção;
- A concepção do software com abordagens sobre a engenharia de requisitos, como ocorre a viabilidade e o desenvolvimento do processo de requisitos do software, destacando o estudo para análise dos principais requisitos: do usuário, funcionais, do sistema e não funcionais.

Fonte: ClipArt

- A engenharia de software se ocupa de todos os aspectos necessários para produzir sistemas de software, dos estágios iniciais de conhecimento do negócio, da análise, especificação, modelagem, construção, testes, entrega, suporte até a manutenção.

Definição de engenharia de software

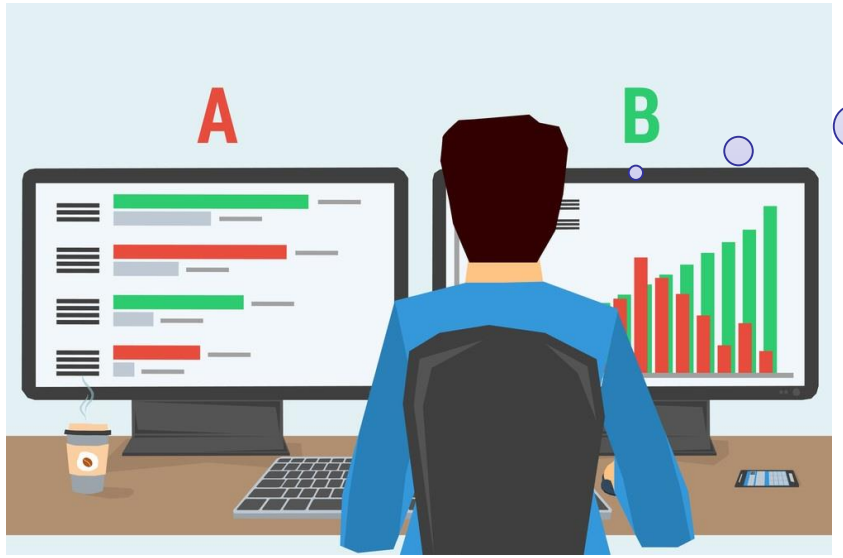
- A engenharia de software projeta e constrói o produto software de computador.
- Abrange programas que executam em computadores de qualquer tamanho, arquitetura ou volume de processamento.
- Combinam-se dados de vários tipos e as informações são apresentadas em diversas formas: impressas, virtuais, textos dos mais variados tipos, vídeos, áudios, imagens em 2D e 3D.

- A engenharia de software usa o conhecimento e resultados de diversas áreas, fornece outros problemas de estudo, bem como auxilia na resolução de problemas. “O mundo moderno não poderia existir sem o software. Infraestruturas e serviços nacionais são controlados por sistemas computacionais, e a maioria dos produtos elétricos inclui um computador e um software que o controla” (Sommerville, 2011).

Relação das engenharias de software e de sistemas

- A **engenharia de sistemas** é uma disciplina presente em muitos projetos de desenvolvimento de software.
- A engenharia de sistemas focaliza diversos elementos que, integrados, dão suporte ao software. O software, no entanto, é o principal elemento.
- A engenharia de sistemas tem a visão voltada para a interface e ligação entre os elementos.

- Os elementos que compõem os sistemas computacionais são: software, hardware, pessoas (peopleware), base de dados e redes de computadores.



Qual o caminho para desenvolver software?

Fonte: Gaea. Veja como desenvolver um projeto de *software.* Disponível em: https://gaea.com.br/veja-como-desenvolver-um-projeto-de-software/. Acesso em 05 jan. de 2024.

Engenharia de sistema: Análise de infraestrutura de TI para software

- Em um estúdio de gravações de vídeo e áudio para web, o software WalkStudio é responsável por todo o controle operacional. Anteriormente à implantação, os testes ocorreram em ambiente restrito e nesta fase será feito o teste de integração.
- O software deverá ser implantado em uma rede local de computadores e o bom desempenho deve ser medido de acordo com a taxa de transferência de dados ao servidor SGDB.

Ambiente operacional:

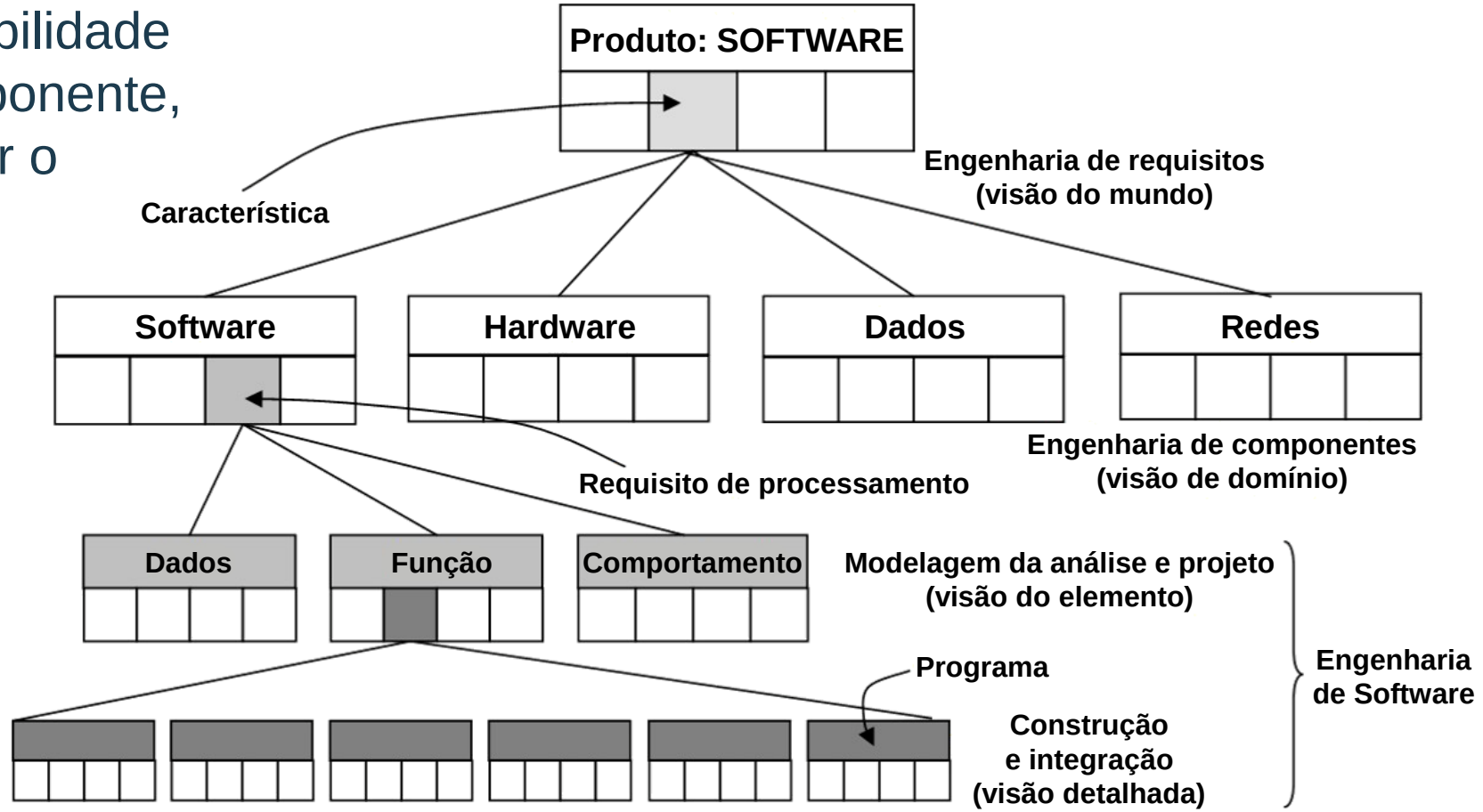
1. O software é processado na estação e a base de dados está no servidor SGBD.
2. O ambiente é multitarefa, no qual ocorrem múltiplos acessos a funções em múltiplas estações de computadores em tempo real.

Questões a serem levantadas:

1. Que outras aplicações estão sendo processadas no servidor?
2. Medir o desempenho da CPU para a chamada de dados.
3. Medir se a banda de rede está compatível com o tráfego de dados no pior caso.

Características do software: Hierarquia de análise do produto software

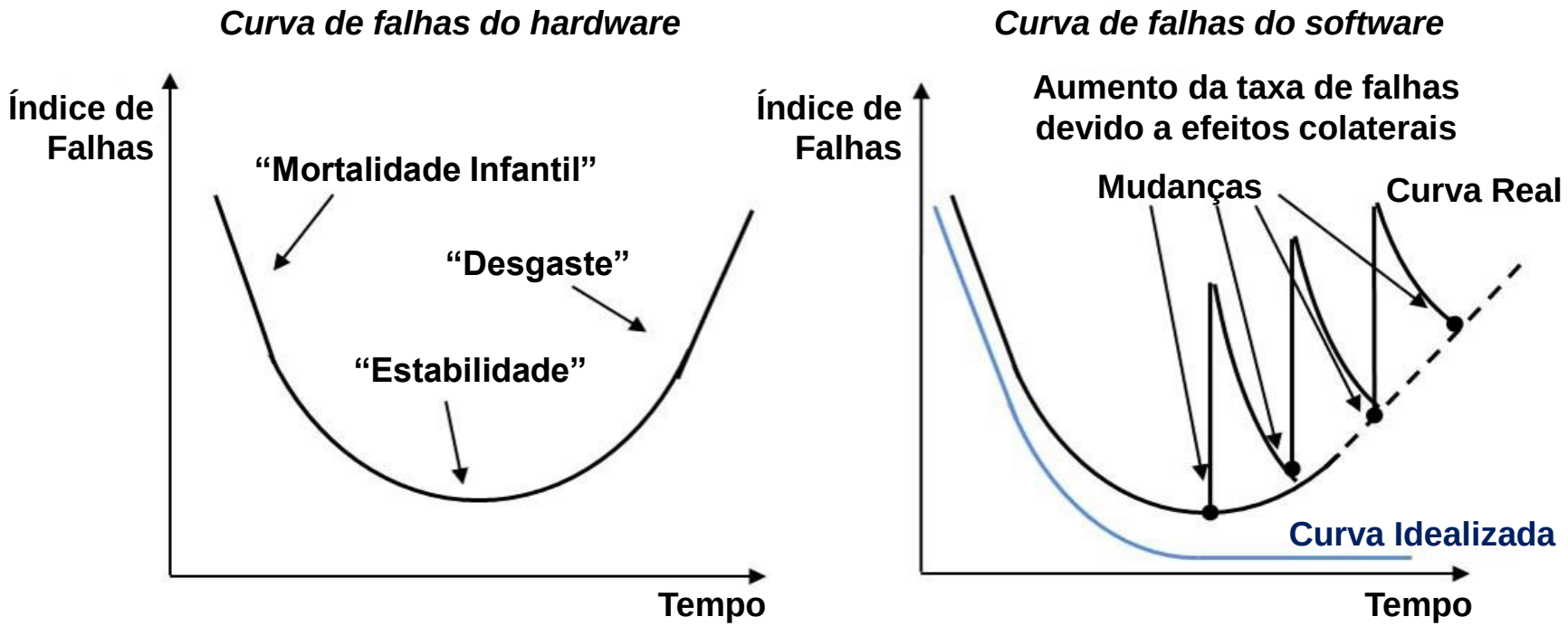
- A construção do software é conduzida por uma série de eventos que evoluem junto com as prioridades de requisitos do software.
- A hierarquia de análise conduz à construção do software.
- O objetivo é garantir a disponibilidade dos requisitos para cada componente, que permitam codificar e testar o software em construção.



Fonte: adaptado de: Moreno (2020); Pressman (2002).

Características do software: Dualidade do software com o hardware

- A dualidade do software com o hardware permite a construção de sistemas computacionais cada vez mais avançados. Contudo, a real função do computador está no software.
- O hardware é manufaturado e se desgasta. “O software não se desgasta mas se deteriora” (Pressman, 2011) e é desenvolvido por processos de engenharia.
- As mudanças do software ocorrem devido a: falhas, implementação de novas funções ou por requisição do usuário.



Fonte: Moreno (2018), adaptado de Pressman (2011).

Características do software: Acompanhamento das mudanças

- No acompanhamento da evolução do software devido às mudanças implementadas, estas são registradas por meio de versões e releases e quando se deterioram o software é reestruturado.
- Versões – registros do software feitos sobre as mudanças que ocorrem quando o software está em desenvolvimento.
- Release (lançamento) – registro da versão que é liberada para o usuário.

Após várias mudanças causadas, o software deve ser reestruturado, o que significa:

- Fazer limpeza dos dados.
- Fazer limpeza dos códigos redundantes.
- Atualizar hardware.
- Atualizar com novas versões o sistema operacional e as linguagens de programação.
- Gerar novos algoritmos.
- Adaptar de forma correta as antigas e novas funcionalidades com base em uma nova arquitetura.

Processos de software

- O processo é um diálogo no qual o conhecimento, que deve se transformar em software, é reunido e embutido no software (Pressman, 2002).

- Um modelo de processo de software é uma representação simplificada de um processo de software. Cada modelo representa uma perspectiva particular de um processo e, portanto, fornece informações parciais sobre ele (Sommerville, 2011).

Os modelos de processo de software mais conhecidos são:

- Modelos de processos tradicionais: Cascata, Balbúrdia, Prototipagem, Incremental, RAD e Espiral;
- Processo Unificado: RUP e Praxis;
- Modelos de processos pessoal e de equipe: PSP e TSP.

Processos de software: Modularidade e componentes

- A modularidade consiste em dividir o sistema de software em módulos ou componentes, que trabalham em conjunto para atingir um determinado objetivo.

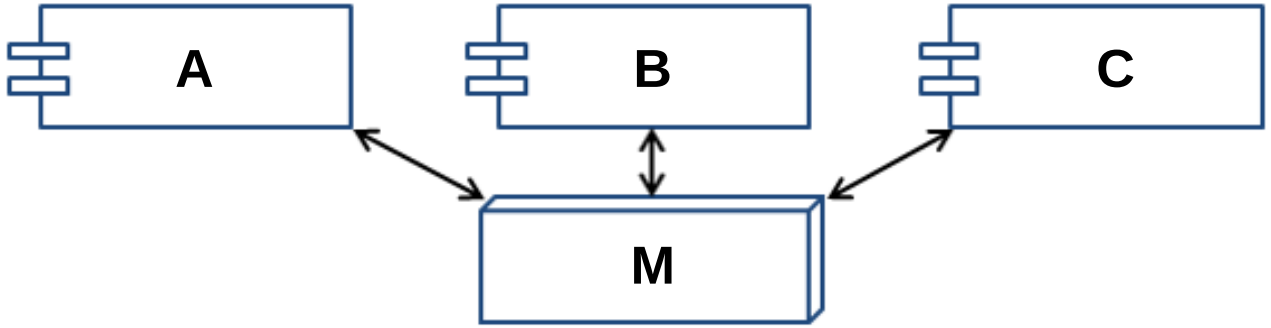
- O módulo ou o componente de software são blocos isolados, que independem de outras partes do sistema, possuem endereçamento próprio e permitem manutenções isoladas sem que estas afetem outras partes do sistema.

    - A diferença básica entre um módulo e um componente de software está associada ao tamanho, complexidade ou qualidade exigida.

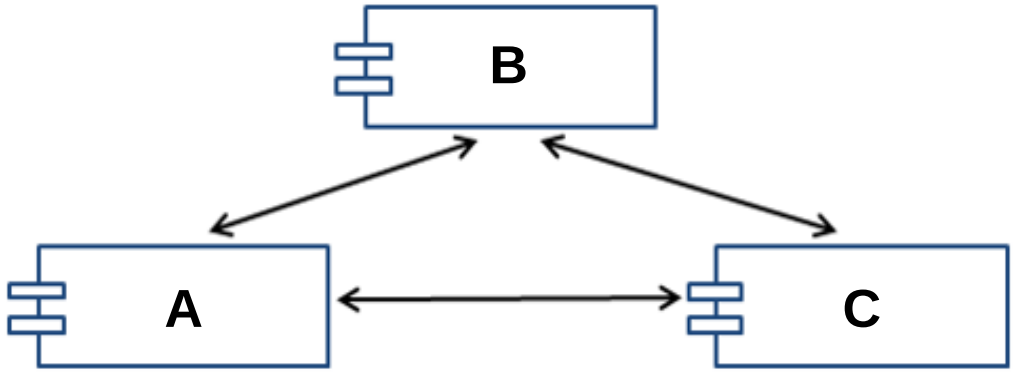
    - Veja bem: Você pode ter dois componentes de software, um que faça cálculo de folha de pagamento e outro que gere relatórios. Contudo, os dois componentes podem ser integrados em um único módulo que faça essas duas operações.

Processos de software: Acoplamento e coesão

- O que se deseja atingir com a modularidade é: baixo acoplamento e alta coesão.
- No Modelo 1, os módulos A, B e C possuem uma alta dependência do módulo M, um índice alto de acoplamento. Se o módulo M falha, compromete os módulos A, B e C. Não é bom.



  - No Modelo 2, os módulos A, B e C possuem alta coesão. Mesmo que um dos módulos apresente falha, os dois outros garantem a troca de mensagens. É bom.
  - Modelo 2: Alta coesão.



Engenharia de requisitos

O software começa aqui.

- A concepção do projeto de software constitui-se dos procedimentos para criar o documento de requisitos do software.

Nesse documento se definem os grupos de requisitos:

A engenharia de requisitos contempla as principais atividades e práticas do desenvolvimento do produto software:

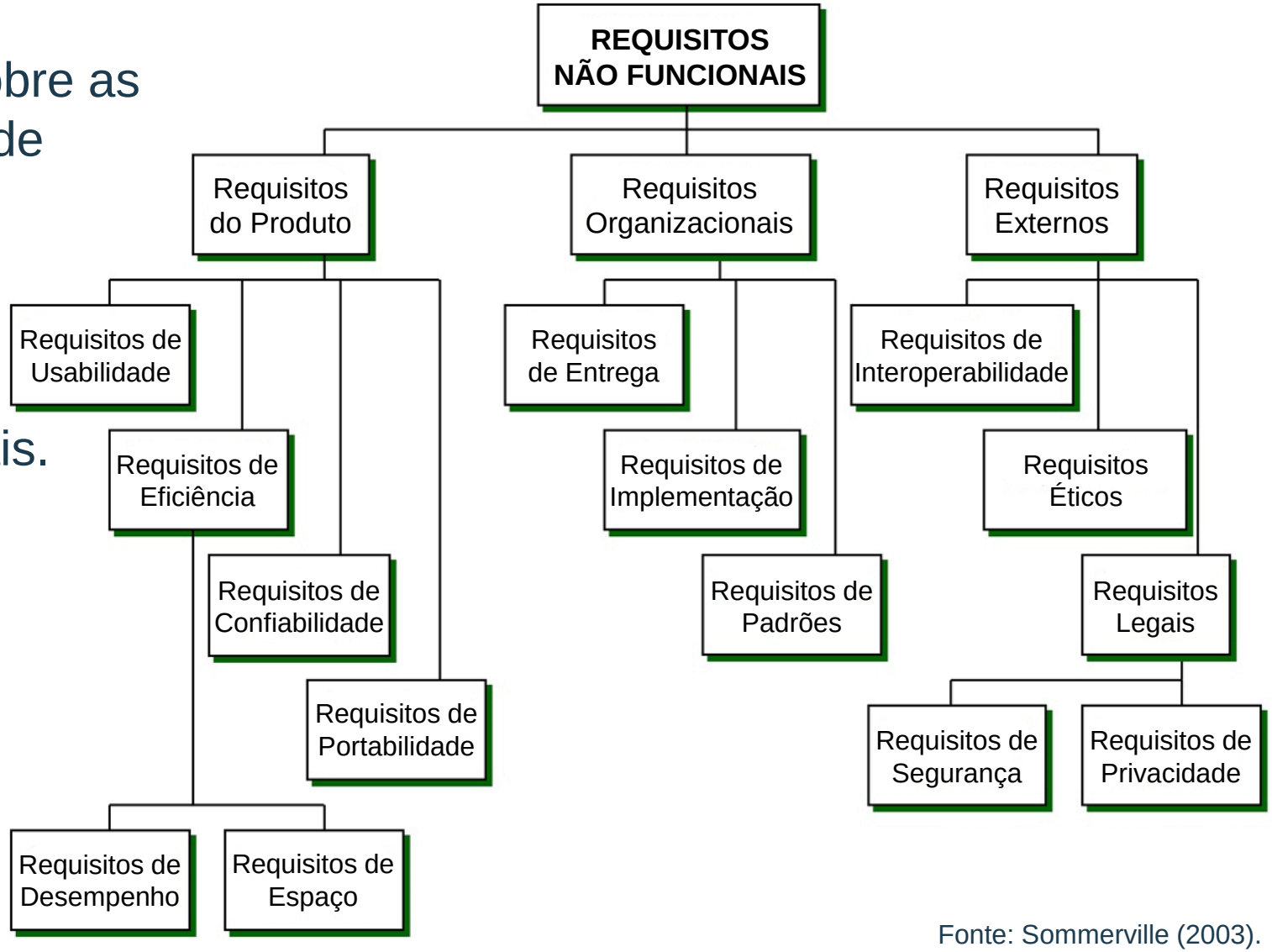
Principais grupos de requisitos
▪ Requisitos do Usuário; ▪ Requisitos do Sistema; ▪ Requisitos Funcionais; e ▪ Requisitos Não Funcionais.

Atividades do processo de engenharia de requisitos do sistema de software
▪ Estudo da Viabilidade; ▪ Elicitação; ▪ Análise; ▪ Especificação; ▪ Modelagem; ▪ Validação.

Engenharia de requisitos: do Usuário (RU) e Não Funcionais (RNF)

- Requisitos do Usuário (RU): São declarações em linguagem natural, formulários e diagramas simples, sobre as funções e restrições que o sistema de software deve fornecer.
- Requisitos Não Funcionais (RNF): Determinam a qualidade do software. Na figura são mostrados os tipos de Requisitos Não funcionais.

Tipos de Requisitos Não Funcionais (RNF)



Fonte: Sommerville (2003).

Engenharia de requisitos: Funcionais (RF) e do Sistema (RS)

- Requisitos Funcionais (RF): São descrições detalhadas dos RU com a especificação das funcionalidades do software.

- Requisitos do Sistema (RS): São descrições detalhadas dos RU com foco no sistema, com uma especificação completa dos componentes do sistema.

Requisito Funcional (RF)	Especificação
RF01	Chamada de menu: Relatórios – deverá exibir o menu de relatórios disponíveis.
RF01.1	Função: Relatório de compras – relatório com as compras efetuadas no período desejado. Exibindo: Fornecedor, produto comprado, quantidade e valor.
RF01.2	Função: Relatório de pagamentos efetuados – relatório com os pagamentos efetuados a fornecedores no período desejado. Exibindo: Fornecedor, produto comprado, quantidade e valor pago.

Requisito do Sistema (RS)	Especificação
RS01	Computador cliente, modelo PC com médio desempenho.
RS02	Sistema operacional do computador cliente – Windows. Ver RS01.
RS03	Navegador para internet. A aplicação irá funcionar em uma rede intranet.

Interatividade

Com base na engenharia de software, qual alternativa completa a lacuna corretamente?

“À medida que se incluem mudanças no software: correções, adaptações ou implementação de novos recursos, o software ao longo do tempo começa a travar, perder dados e a ter queda de desempenho. Estima-se que esses problemas ocorram devido a _____”.

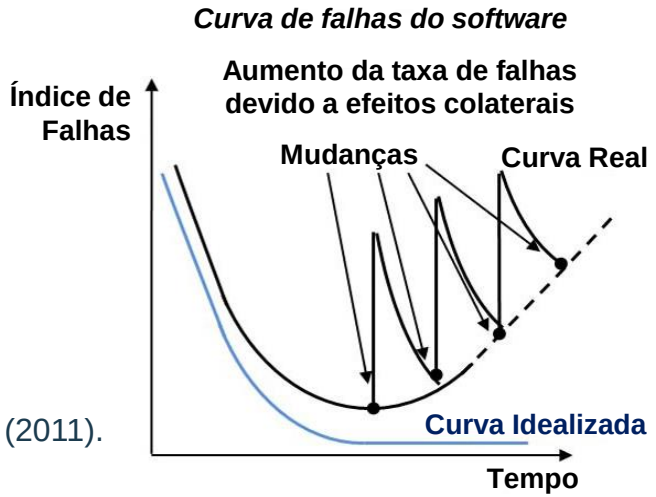
a) uma exigência de atualização de hardware com maior desempenho.
b) uma manutenção periódica precária do sistema de software.
c) um número de falhas que aumenta à medida que são feitas novas versões do software.
d) uma necessidade de controle da qualidade do sistema de software em relação à tecnologia.
e) uma nova geração de sistema operacional que compromete as mudanças necessárias.

Resposta

Com base na engenharia de software, qual alternativa completa a lacuna corretamente?

“À medida que se incluem mudanças no software: correções, adaptações ou implementação de novos recursos, o software ao longo do tempo começa a travar, perder dados e a ter queda de desempenho. Estima-se que esses problemas ocorram devido a _____”.

a) uma exigência de atualização de hardware com maior desempenho.
b) uma manutenção periódica precária do sistema de software.
c) um número de falhas que aumenta à medida que são feitas novas versões do software.
d) uma necessidade de controle da qualidade do sistema de software em relação à tecnologia.
e) uma nova geração de sistema operacional que compromete as mudanças necessárias.

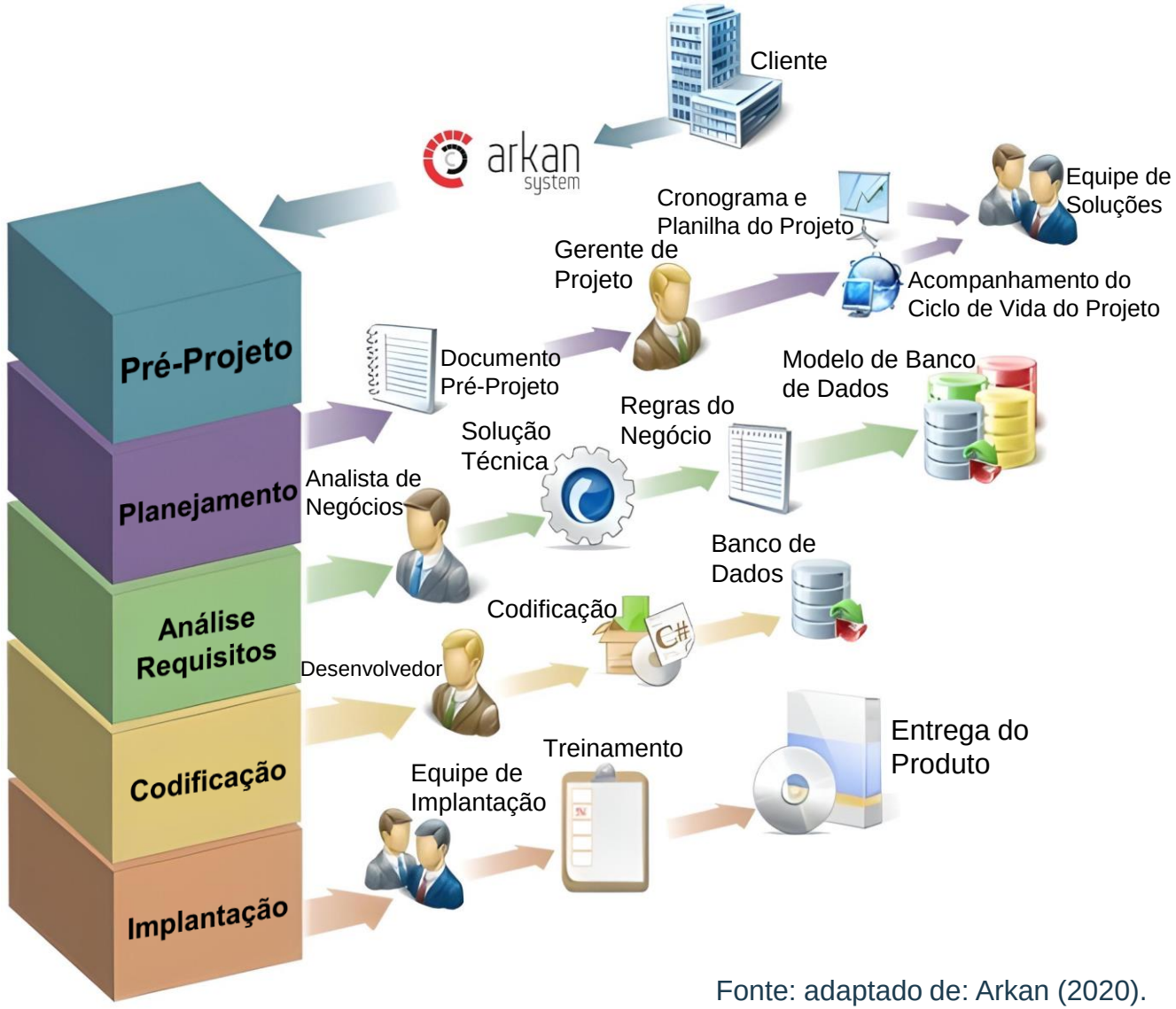


Fonte: adaptado de: Pressman (2011).

2. Ciclo de vida do software

- Um exemplo de ciclo de vida do software é apresentado pela empresa Arkan, como mostra o modelo.

- O ciclo de vida do desenvolvimento de software está embasado na qualidade e abrange várias fases, desde a concepção, design, implementação, testes, implantação, manutenção e suporte.



Fonte: adaptado de: Arkan (2020).

NBR ISO/IEC 12207 – Processos do ciclo de vida do software

- A NBR ISO/IEC 12207 – Processos do ciclo de vida do software – fornece as diretrizes para os processos envolvidos no desenvolvimento de software.

Estão organizados em três categorias, como mostra o framework:

<table>
<tr><th colspan="2">1. Processos fundamentais</th><th>2. Processos de apoio</th></tr>
<tr><td colspan="2">1.1 Aquisição</td><td>2.1 Documentação</td></tr>
<tr><td colspan="2">1.2 Fornecimento</td><td>2.2 Gerência de Configuração</td></tr>
<tr><td rowspan="6">1.3 Desenvolvimento</td><td rowspan="3">1.4 Operação</td><td>2.3 Garantia da Qualidade</td></tr>
<tr><td>2.4 Verificação</td></tr>
<tr><td>2.5 Validação</td></tr>
<tr><td rowspan="3">1.5 Manutenção</td><td>2.6 Revisão Conjunta</td></tr>
<tr><td>2.7 Auditoria</td></tr>
<tr><td>2.8 Resolução de Problema</td></tr>
<tr><th colspan="3">3. Processos organizacionais</th></tr>
<tr><td colspan="2">3.1 Gerência</td><td>3.3 Infraestrutura</td></tr>
<tr><td colspan="2">3.2 Melhoria</td><td>3.4 Treinamento</td></tr>
</table>

Fonte: NBR ISO/IEC 12207 (1998).

Gerência de configuração e mudanças de software

- O gerenciamento de configuração define como registrar e processar as mudanças do sistema, como relacioná-los aos componentes e os métodos utilizados (Sommerville, 2011).

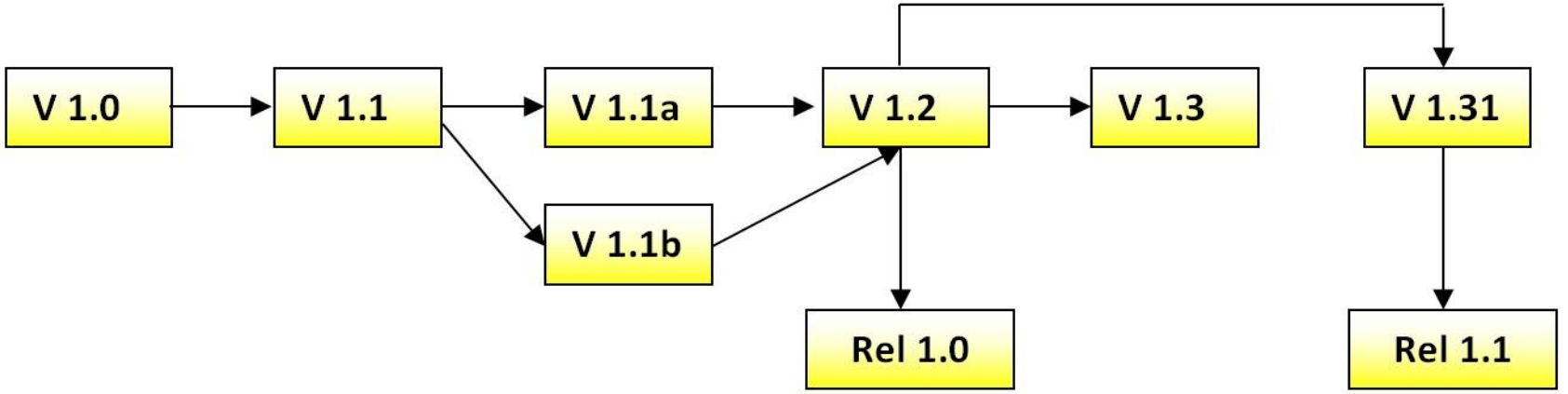
Atividades de configuração de software	
Gerenciamento de mudanças	Registro histórico das mudanças aplicadas no sistema.
Gerenciamento de versões	Registro e acompanhamento das versões do sistema.
Construção de sistemas	Processo de compilar e ligar componentes de software em um programa para uma configuração específica.
Gerenciamento de releases	Registro e acompanhamento das versões liberadas para o cliente.

Fonte: Sommerville (2011).

Gerenciamento de versões e releases

Técnicas básicas de numeração e identificação da versão	
Numeração de versões	Versão 3.21 – (3) identifica mudança de estrutura, (2) indica inserção de funcionalidade e (1) indica que houve implementação de uma mudança.
Identificação por atributos	Versão tec01_1: (tec) representa teclado, (01) inserção de mapa de caracteres e (_1) implementação de mudança.
Identificação orientada a mudanças	car_04: (car) Carlos requisita uma quarta mudança.

- Versão: São registros de testes ou mudanças e não são liberadas para o cliente.
- Release: É o registro da versão do sistema de software liberada para o cliente.



Fonte: Moreno (2020).

Projeto e construção do sistema de software

Fonte: adaptado de: Pressman (2002).

- A engenharia de software é uma tecnologia em camadas e que deve estar fundamentada em um comprometimento organizacional com a qualidade (Pressman, 2002).
- Qualidade: Determina padrões e normas a serem aplicadas pela engenharia de software.
- Processo: Forma a estrutura básica do projeto que mantém integradas as camadas da tecnologia.
- Método: Conjunto de procedimentos, regras e operações que fornecem uma técnica que permite chegar a uma determinada meta, fim ou conhecimento.
- Ferramentas: fornece apoio automatizado ou semiautomatizado para os processos e para os métodos.

Gerenciamento de riscos do projeto

- O risco pode ser conceituado como a combinação da probabilidade de um evento e de suas consequências.
- O gerenciamento dos riscos do projeto tem por objetivo aumentar a probabilidade e/ou o impacto dos riscos positivos e diminuir a probabilidade e/ou o impacto dos riscos negativos, a fim de otimizar as chances de sucesso do projeto (PMBOK®, 2017).

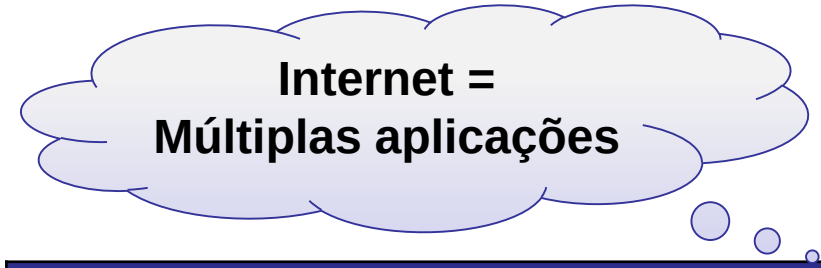
Fonte: ClipArt

Processos de gerenciamento dos riscos do projeto

1. Planejar o gerenciamento dos riscos;
2. Identificar os riscos;
3. Realizar a análise quantitativa dos riscos;
4. Planejar as respostas aos riscos;
5. Implementar respostas aos riscos;
6. Monitorar os riscos.

Desenvolvimento de sistemas de informação para internet

- As páginas da web recuperadas por um browser constituem o software que incorpora instruções executáveis (CGI, HTML, Pearl, Java e outras) e dados (hipertextos e uma variedade de formatos visuais e de áudio).
- Desses recursos tecnológicos, surgem os sistemas e aplicações baseadas na web, as WebApps, que são pequenas aplicações embarcadas na internet.
- No mundo dos negócios, a amplitude dessa área está classificada pelo termo Comércio Eletrônico (e-commerce), que é estruturado basicamente pelas tecnologias: B2B (business to business), B2C (business to consumer) e C2C (consumer to consumer).

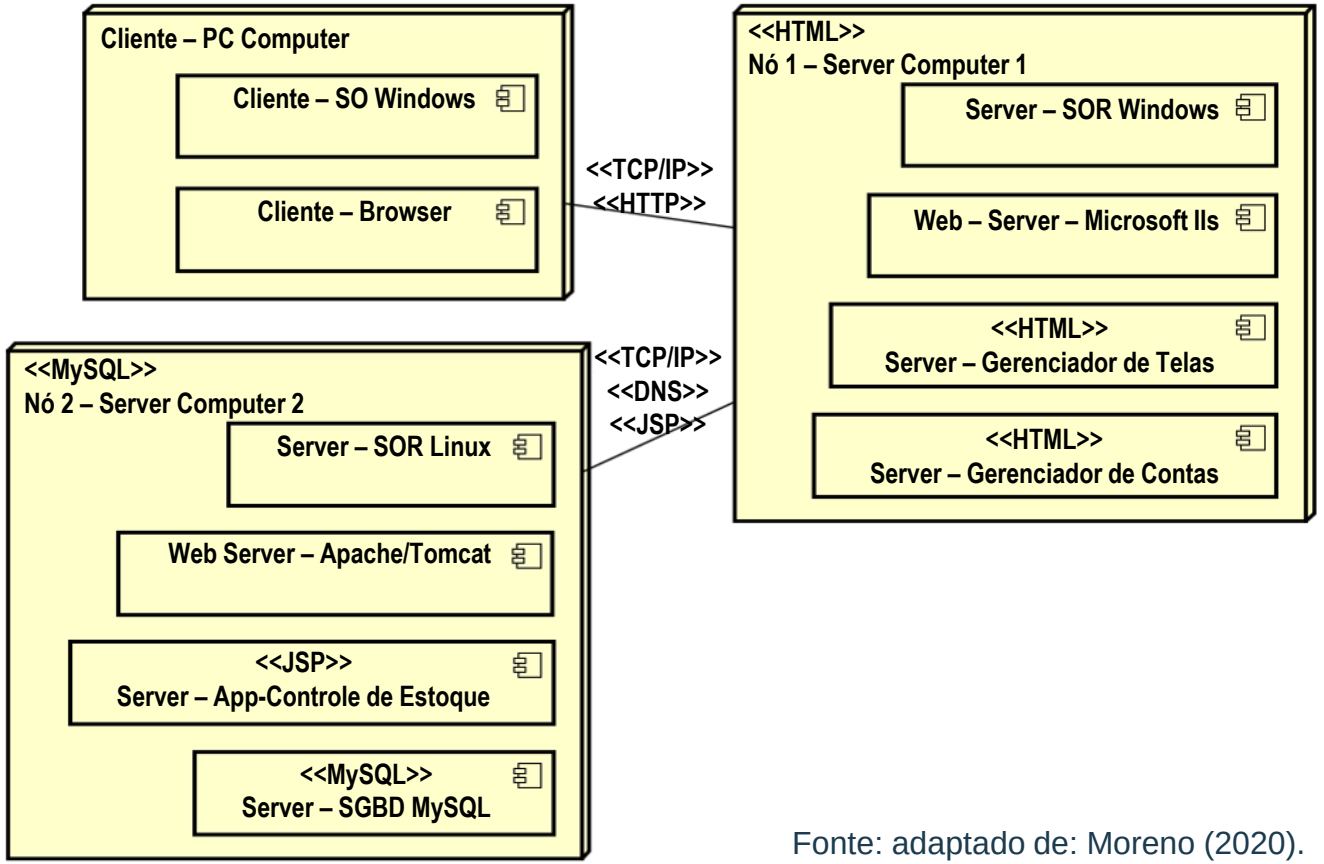


Saiba automatizar as coisas pela internet

- DIGICOMP. Internet das coisas: como ela otimiza os recursos na Indústria?. Digicomp Engenharia e Tecnologia. Disponível em: https://digicomp.com.br/internet-das-coisas-como-ela-otimiza-os-recursos-na-industria/. Acesso em: 12 jan. 2024.

Desenvolvimento de software para internet

- A internet tem como base uma arquitetura servidor/cliente em ambiente distribuído. Observe a arquitetura abaixo.
- O cliente por meio do browser ou de uma aplicação específica (app) recebe documentos em vários formatos que podem ser manipulados por plug-ins ou helpers, sem ter que mudar o browser ou a aplicação.
- O servidor apenas entrega documentos e não se preocupa com as interfaces do usuário ou formato dos documentos.



Fonte: adaptado de: Moreno (2020).

Interatividade

A engenharia de software é uma tecnologia em camadas. Uma dessas camadas determina “incluir um amplo conjunto de tarefas que abrangem gestão de equipes, análises de requisitos, projeto, construção de programas, teste e manutenção”. Essas atividades são características de qual camada da engenharia de software?

a) Ferramentas.
b) Garantia de qualidade do software.
c) Interdependência de sistemas.
d) Métodos.
e) Usabilidade.

Resposta

A engenharia de software é uma tecnologia em camadas. Uma dessas camadas determina “incluir um amplo conjunto de tarefas que abrangem gestão de equipes, análises de requisitos, projeto, construção de programas, teste e manutenção”. Essas atividades são características de qual camada da engenharia de software?

a) Ferramentas.
b) Garantia de qualidade do software.
c) Interdependência de sistemas.
d) Métodos.
e) Usabilidade.

Fonte: adaptado de: Pressman (2002).

3. Organização e o processo de desenvolver software

- A organização e o processo de desenvolver software têm como base modelos e técnicas provadas como eficazes e eficientes na construção do software.

Fonte: ClipArt

Os modelos e técnicas que se destacam estão em:

- Joint Application Development (JAD) (em português: Desenvolvimento de Aplicação Conjunta): Método formal criado pela IBM em 1977 para criação ou revisão do processo e que ainda se faz presente nos tempos atuais.
- Rational Unified Process (RUP) (em português: Processo Unificado Racional): modelo de processo unificado. Esse modelo percorre todas as etapas do ciclo de vida de desenvolvimento do software.
- Fatores e métricas da qualidade do software: Conceito de métricas e medidas, que acompanham o progresso de qualidade do software.

Joint Application Development (JAD)

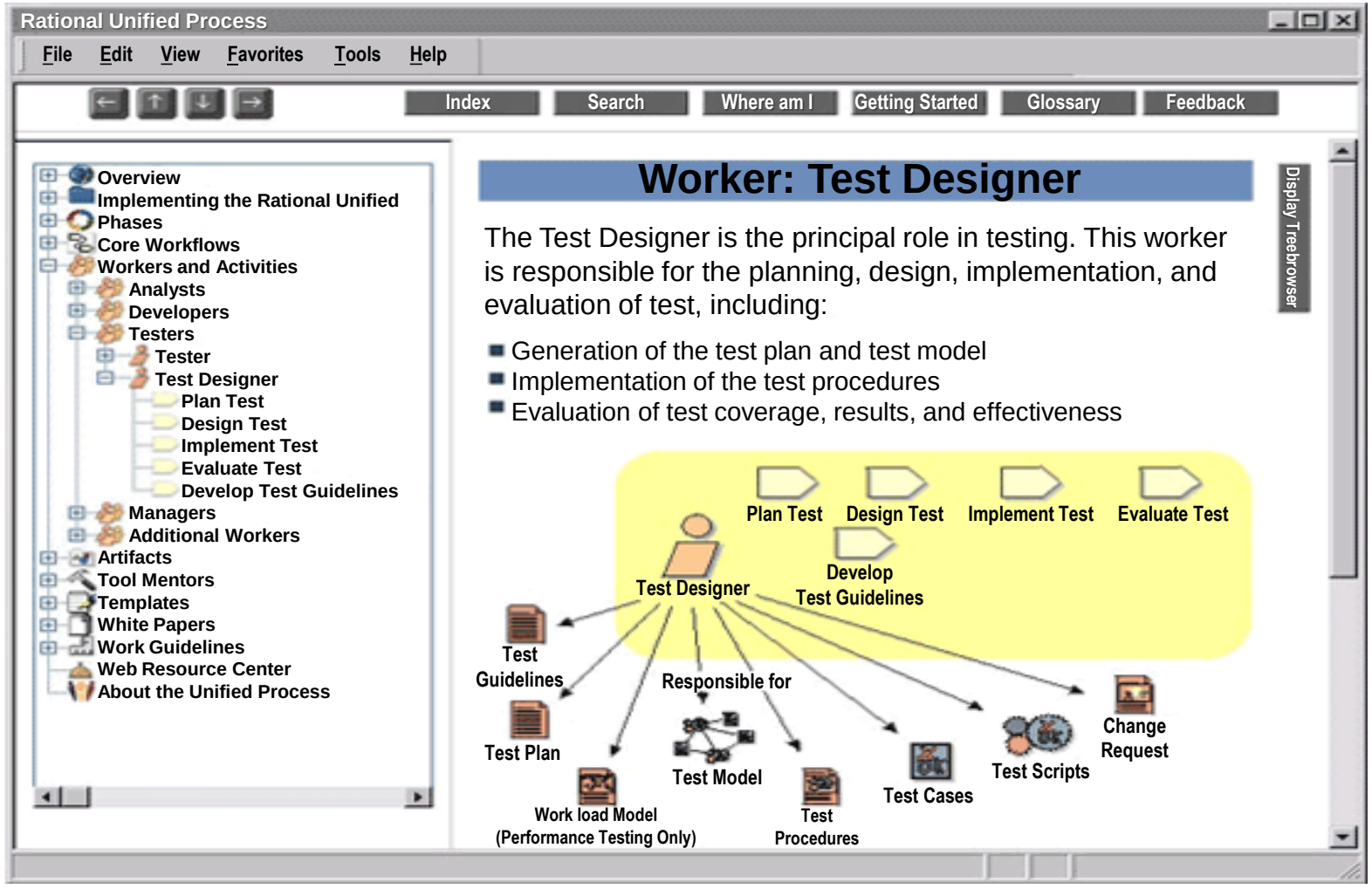
As sessões JAD são usadas no setor de desenvolvimento de software e de acordo com Fournier (1994) os participantes das sessões JAD são:

- Executivo patrocinador: Fornece diretrizes das principais metas e objetivos do projeto;
- Gerência funcional: Representante do usuário que descreve práticas atuais do negócio;
- Representantes do sistema: Pessoas com conhecimentos técnicos das operações;
- Líder da sessão: Responsável pelo controle das comunicações entre os participantes.
- Secretário: Utiliza ferramentas e técnicas documentais e de modelagem para os registros.

- Essas sessões facilitadas são focadas em reunir os especialistas em assuntos de negócio e a equipe de desenvolvimento para coletar requisitos e melhorar o processo de desenvolvimento de software (PMBOK®, 2017).

Rational Unified Process (RUP)

- O RUP - Rational Unified Process (Processo Unificado da Rational) é um processo proprietário de engenharia de software criado pela Rational Software Corporation, representado pela IBM ®, com o nome IRUP é uma abreviação de IBM Rational Unified Process.



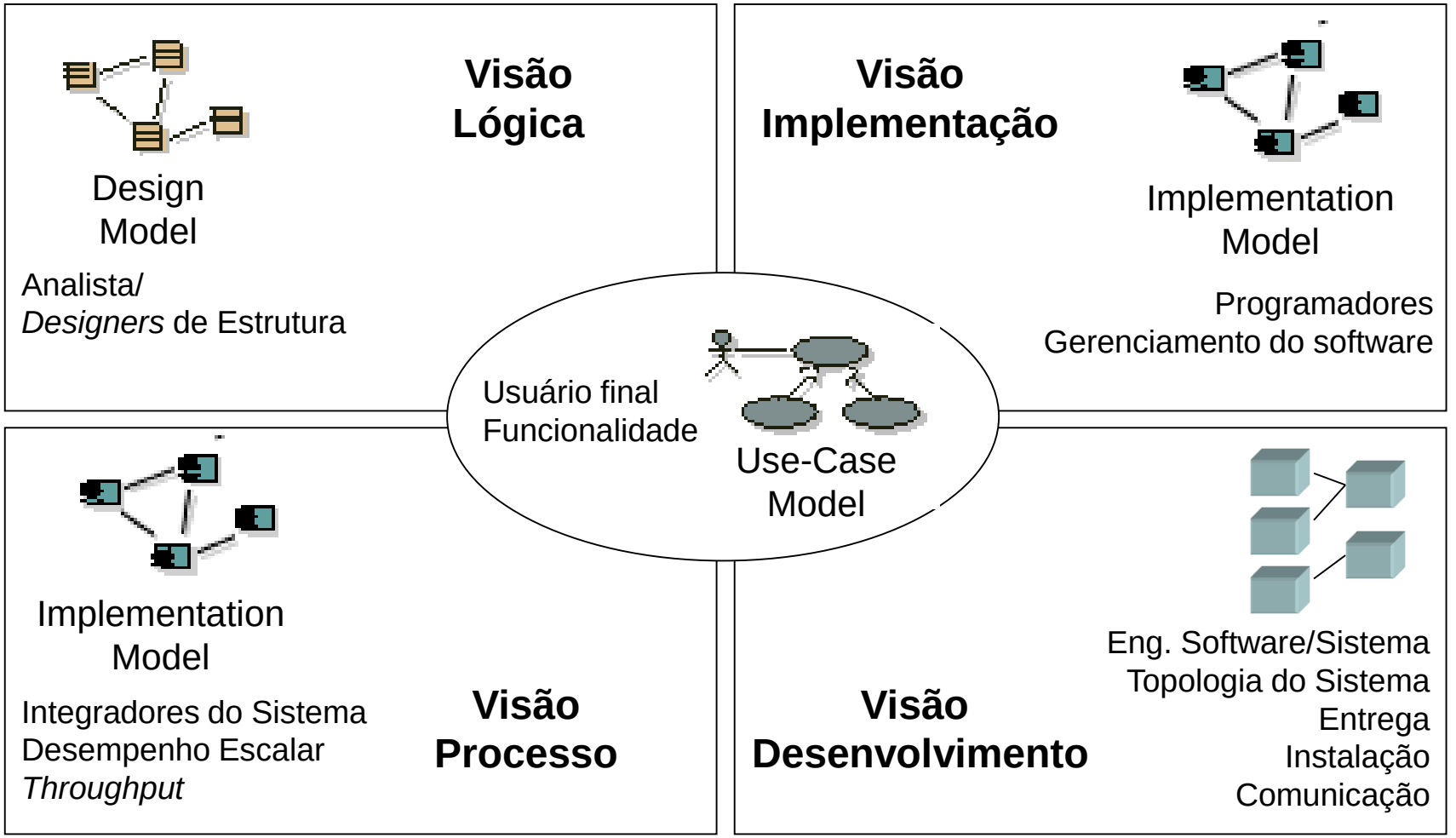
Fonte: adaptado de: Kruchten (2000).

Rational Unified Process (RUP): Características

- Desenvolvimento iterativo – atividade revisional contínua devido a mudanças frequentes que podem ocorrer no ciclo de desenvolvimento.
- Gestão de requisitos – o RUP é conduzido por casos de uso.
- Arquitetura baseada em componentes – é desenhada e documentada usando UML. O RUP dá suporte à construção do sistema com o foco em uma arquitetura executável, logo nas primeiras fases do projeto.
- Uso de software em modelo visual – o ambiente operacional é desenhado, desenvolvido, distribuído e mantido como uma ferramenta de software.
    - Verificação contínua da qualidade – É distribuído on-line, utilizando tecnologia web e está sob o formato eletrônico e pode ser configurado de acordo com as necessidades de cada organização.
    - Gestão e controle de mudanças do software – definição de métodos para controle e monitoração do ciclo de desenvolvimento.

Rational Unified Process (RUP): Orientado por casos de uso

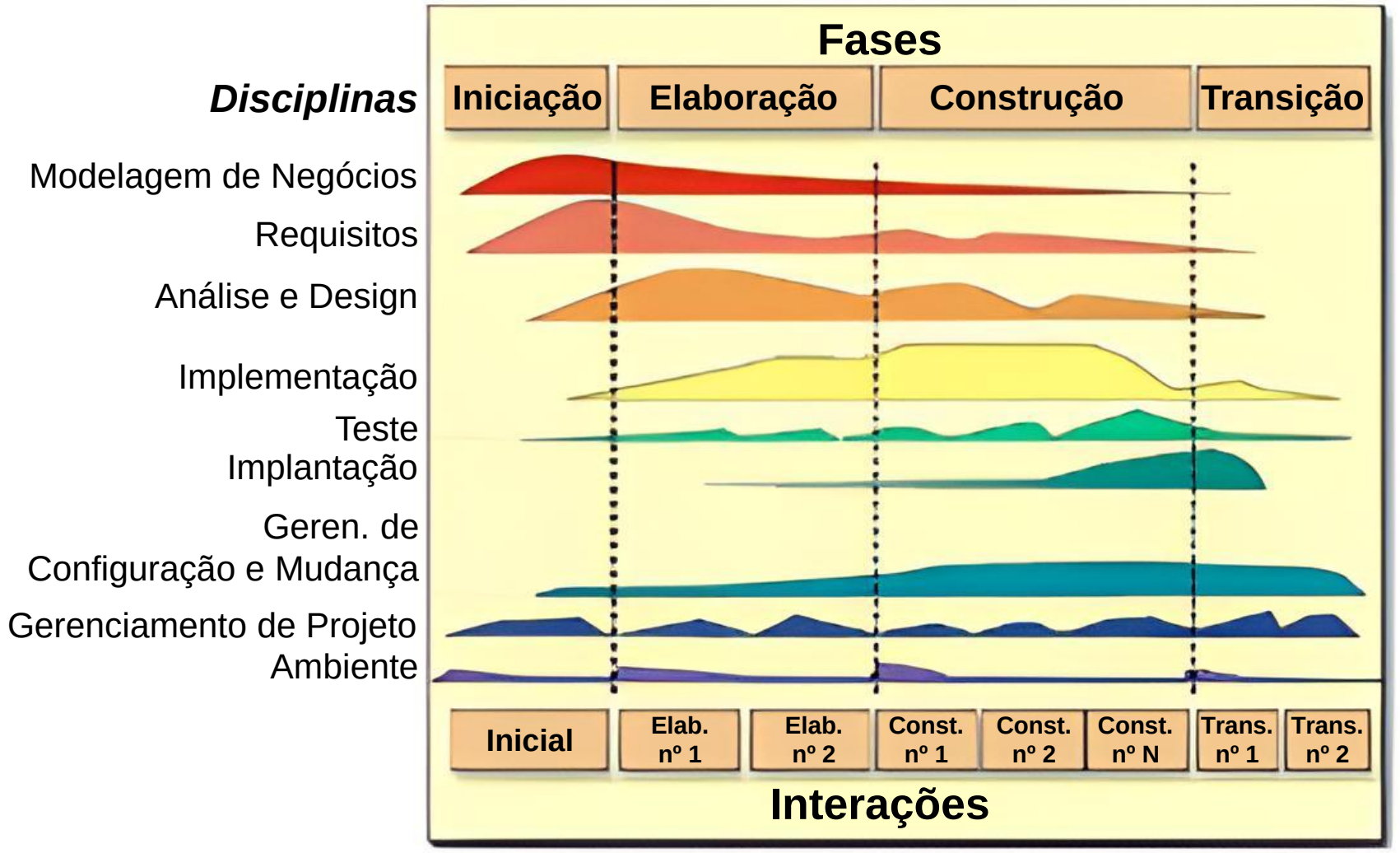
- O RUP é orientado por casos de uso.
- Como o modelo, mostra que o início se dá pela definição do Caso de Uso, que a partir daí permite abstrair conhecimentos das atividades de Processo, Lógica, Desenvolvimento e Implementação.



Fonte: adaptado de: Kruchten (2000).

Rational Unified Process (RUP): Estrutura dinâmica e estática

- A estrutura dinâmica, na horizontal, é representada pelas fases do ciclo de desenvolvimento do software.
- A estrutura estática, na vertical, é representada por nove disciplinas (workflows), que correspondem às atividades do RUP.



Fonte: adaptado de: Kruchten (2000).

Fatores, atributos e métricas da qualidade do software

- No contexto de qualidade de software, é preciso saber que a qualidade é uma mistura complexa de fatores que variam para cada aplicação e com o que os clientes desejam.

Existem três conceitos distintos para determinar a qualidade:

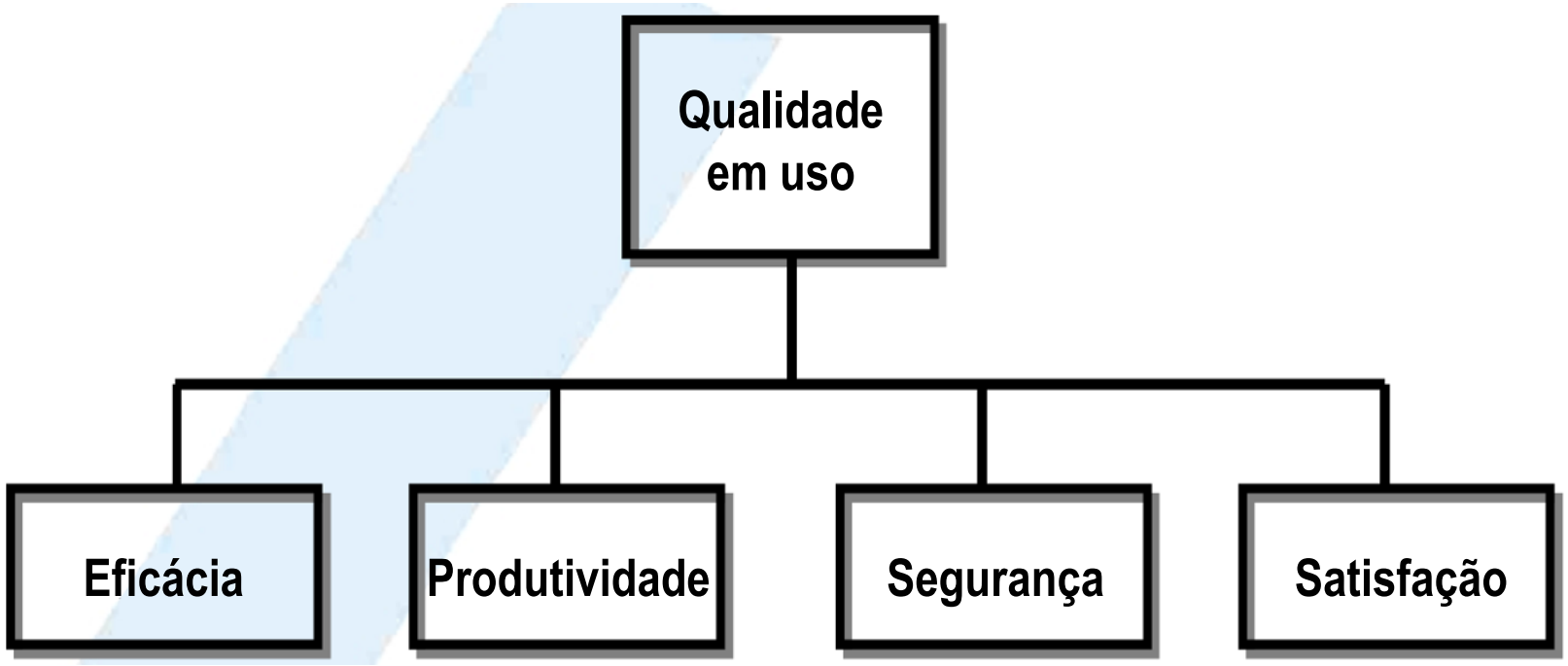
- Fator de qualidade: O conjunto de fatores de qualidade do produto software permite controlar os níveis dos padrões e requisitos que estão sendo implementados.
- Atributo da qualidade: Representa um aspecto particular do produto, que é determinado em um fator de qualidade. Cada fator de qualidade pode ter vários atributos associados a ele.
- Métricas da qualidade: São medidas quantitativas que permitem qualificar um determinado atributo.

Algumas métricas de qualidade do software	
Métrica	**Definição**
Funcionalidade	Avaliação da conformidade com os requisitos do cliente.
Usabilidade	Garante a cada categoria de usuário a interface necessária para navegação.

Fator de qualidade, atributo e métrica: Como usar

- De acordo com a ISO 9126, o fator de qualidade Usabilidade define quatro características (ou atributos): Eficácia, Produtividade, Segurança e Satisfação.

- Cada um dos atributos vai determinar a forma de medição correta, a métrica, para acompanhar e avaliar o processo.



Fonte: adaptado de: NBR ISO/IEC 9126-1 (2003).

Fator de qualidade, atributo e métrica: Mudança de requisitos

Problemas decorrentes de mudanças nos requisitos:

- Rastreamento do código, evolução do sistema, alteração do código e testes.
- Acoplamento de fatores sem relação direta com a mudança, como o banco de dados com a página web. Mudanças isoladas em componentes do sistema podem afetar todo o sistema.
- Aumento no custo da manutenção.

Avaliação de qualidade

Métrica	Definição
Acoplamento e Coesão	Grau de acoplamento que pode afetar atributos externos.
Manutenibilidade	Capacidade do produto de software de ser modificado.
Reusabilidade	Flexibilidade de uso do código.

Interatividade

O modelo de processo Rational Unified Process (RUP) define na dimensão vertical os workflows que formam a estrutura estática, representando os serviços do desenvolvimento. Qual das alternativas abaixo mostra algumas das disciplinas da estrutura estática do RUP?

a) Elicitação, modelagem do negócio, gerenciamento do projeto e construção.

b) Estratégia, tática, operação e suporte.

c) Iniciação, elaboração, construção e transição.

d) Modelagem do negócio, requisitos, implementação e teste.

e) Planejamento, análise, projeto e construção.

Resposta

O modelo de processo Rational Unified Process (RUP) define na dimensão vertical os workflows que formam a estrutura estática, representando os serviços do desenvolvimento. Qual das alternativas abaixo mostra algumas das disciplinas da estrutura estática do RUP?

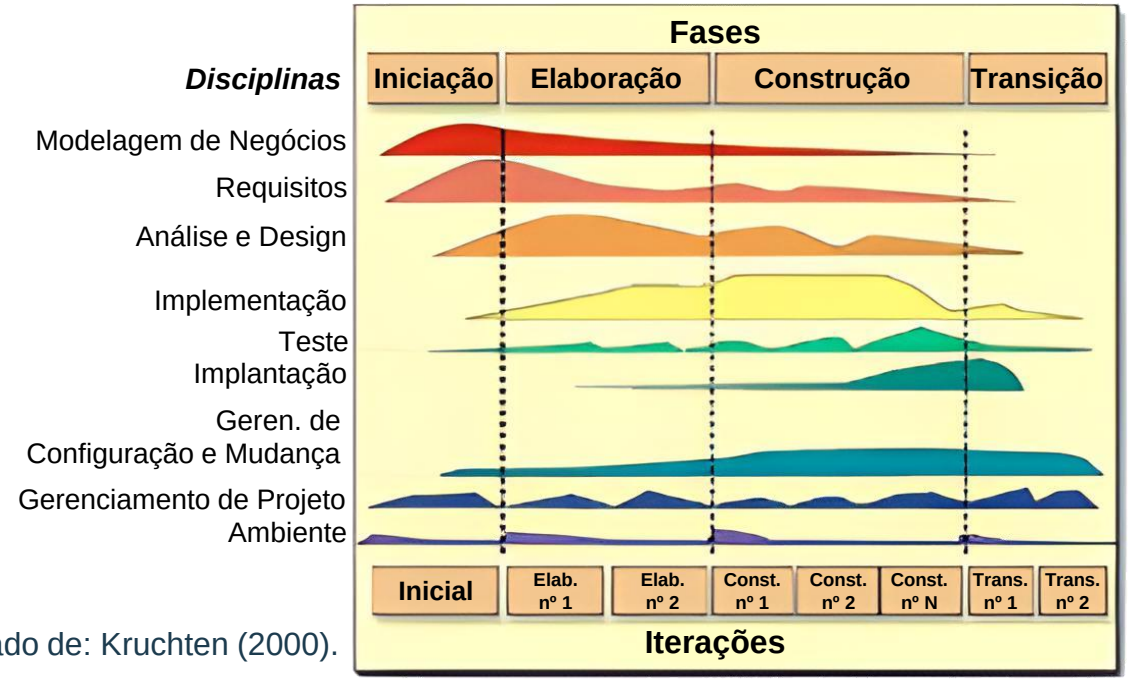
a) Elicitação, modelagem do negócio, gerenciamento do projeto e construção.

b) Estratégia, tática, operação e suporte.

c) Iniciação, elaboração, construção e transição.

d) Modelagem do negócio, requisitos, implementação e teste.

e) Planejamento, análise, projeto e construção.



Fonte: adaptado de: Kruchten (2000).

4. Customização do software

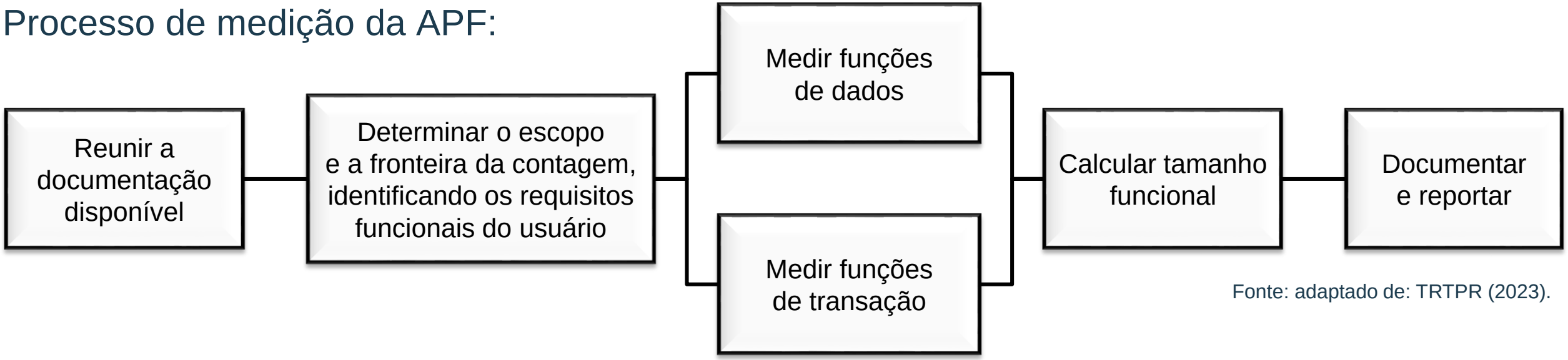
A customização do software é apoiada por metodologias da engenharia de software, que além da customização pode ser utilizada para controle de testes e diagnósticos do software como:

- Análise de Pontos por Função (APF): Serve para customizar, testar e diagnosticar as funções do software, ajudando a eliminar ou mitigar falhas no sistema de software.
- Use-Case Points (UCP): Técnica de contagem usada no desenvolvimento de software para estimar o tamanho e a complexidade de um sistema com base nos casos de uso.
- Testes de Software: Algumas práticas de testes da engenharia de software que auxiliam na customização.

Fonte: ClipArt

Análise de Ponto de Função (APF)

Processo de medição da APF:



Fonte: adaptado de: TRTPR (2023).

De acordo com Pressman (2011) a métrica APF é usada para medir a funcionalidade fornecida por um sistema de software e pode ser empregada para:

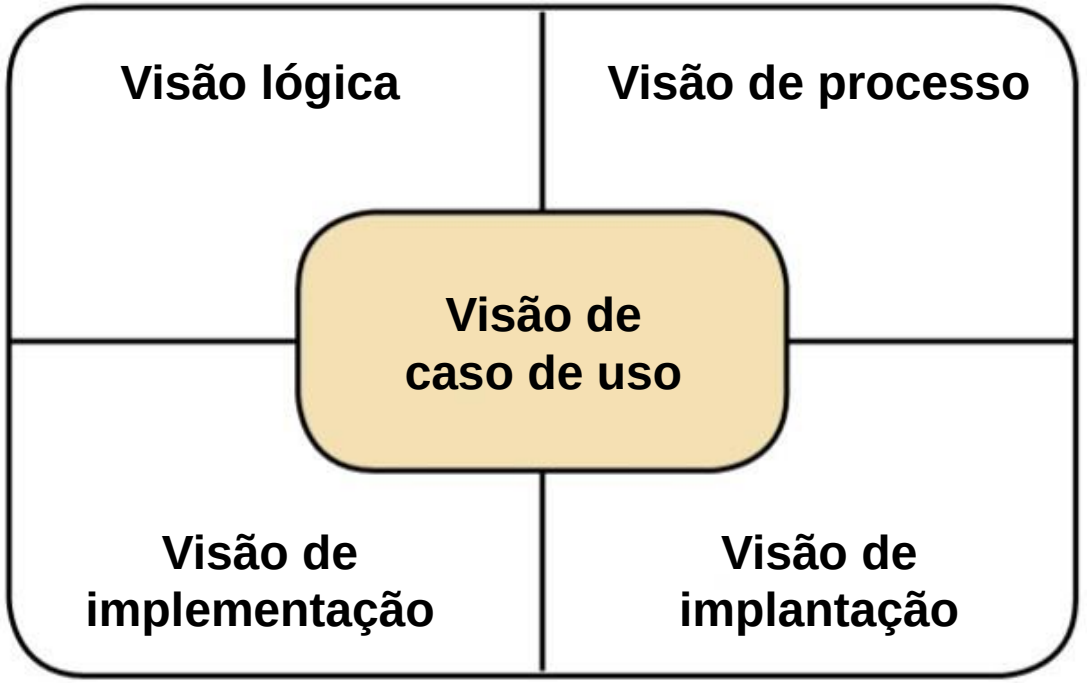
1. Estimar custo ou trabalho necessário para projetar, codificar e testar o software;
2. Prever o número de erros que serão encontrados durante os testes do software;
3. Prever o número de componentes e/ou o número de linhas projetadas de código-fonte no sistema a ser implementado.

Use-case Points (UCP): Conceito

- Use-Case Points (UCP) é uma técnica de contagem usada no desenvolvimento de software para estimar o tamanho e a complexidade de um sistema com base nos casos de uso.

- Em uma abordagem de estimativa com casos de uso, diferente de estimar a contagem por linhas de código (do inglês, lines of code – LOC) ou ponto de função (do inglês, function points - FP), um determinado caso de uso pode precisar de meses de trabalho, enquanto que o caso de uso de outra parte do sistema pode ser implementado em um ou dois dias (Pressman, 2011).

Abstração da análise por casos de uso, de acordo com o RUP e Versolatto:



Fonte: adaptado de: Versolatto (2015).

Use-case Points (UCP): Equações para cálculo de UCP

O cálculo ajustado de UCP (AUCP - Adjusted UCP) é um cálculo complexo que corresponde à estimativa do tamanho e complexidade do sistema com base nos casos de uso, calculado com a equação:

$$AUCP = UUCP + TACF + EF$$

$$UUCP = UUCW + UAW$$

UUCW: Peso para casos de uso.
UUCW: Peso para atores.

Peso: (1) – Simples;
(2) – Mediano;
(3) – Complexo.

$$TCAF = 0,6 + (0,01 \times TFactor\ *)$$

Tfactor: Peso para influência técnica.

$$EF = 1,4 + (-0,03 \times EFactor\ *)$$

Efactor: Peso para influência ambiental.

Use-case Points (UCP): Artefatos produzidos para cálculo de AUCP I

AUCP = UUCP + TACF + EF

Exemplo do cálculo de UUCW:

UUCP = UUCW + UAW

UUCP = 26 + 8 = 34

Tipo de UC	Peso	Nº de Casos de Uso	Total
Simples	1	4	4
Mediano	2	8	16
Complexo	3	2	6
		UUCW	**26**

Tipo de ator	Peso	Nº de Casos de Uso	Total
Simples	1	3	3
Mediano	2	1	2
Complexo	3	1	3
		UAW	**8**

Use-case Points (UCP): Artefatos produzidos para cálculo de AUCP II

AUCP = UUCP + TACF + EF

Exemplo do cálculo de TCAF:

TCAF = 0,6 + (0,01 x TFactor *)

Fator	Requisito	Peso	Influência	TFactor
TF1	Sistema Servidor/Cliente	2	2	4
TF2	Rede de Computadores	2	4	8
TF3	Usabilidade	1	2	2
TF4	Portabilidade	3	5	15
TF5	Segurança	1	4	4
			TCAF	**33**

TCAF = 0,6 + (0,01 + 33) = 0,93

Exemplo do cálculo de EF:

EF = 1,4 + (-0,03 x EFactor *)

Fator	Requisito	Peso	Influência	EFactor
EF1	Programadores (Homens/hora)	3	4	12
EF2	Engenheiros de software (Homens/hora)	2	2	4
EF3	Treinamento dos desenvolvedores	1	3	3
EF4	Treinamento dos usuários	3	5	15
EF5	Recursos	2	5	10
			EF	**44**

EF = 1,4 + (-0,03 + 44) = 0,08

AUCP = UUCP + TACF + EF = 34 + 0,93 + 0,08 = 35,01

Se considerar a produção de 1 UC/20h:
No caso, a produtividade será de 700,2h/sistema

Testes de software

- O teste do software é destinado a mostrar que um programa faz o que é proposto a fazer e para descobrir os defeitos do programa antes do uso (Sommerville, 2011).

- A principal métrica do teste de software é a testabilidade.
- Índice alto de testabilidade: indica a facilidade do software de expor falhas que geram defeitos.
- Índice baixo de testabilidade: indica a dificuldade de identificar falhas que geram defeitos.

- Os testes em sua essência correspondem aos processos de apoio Verificação e Validação (V&V).

Testes de software: Alfa e Beta

- No processo de Verificação e Validação (V&V), quando um software é construído especificamente para um cliente, é normal que ele passe por um teste de aceitação por parte do usuário.

A melhor estratégia são as dos testes Alfa e Beta:

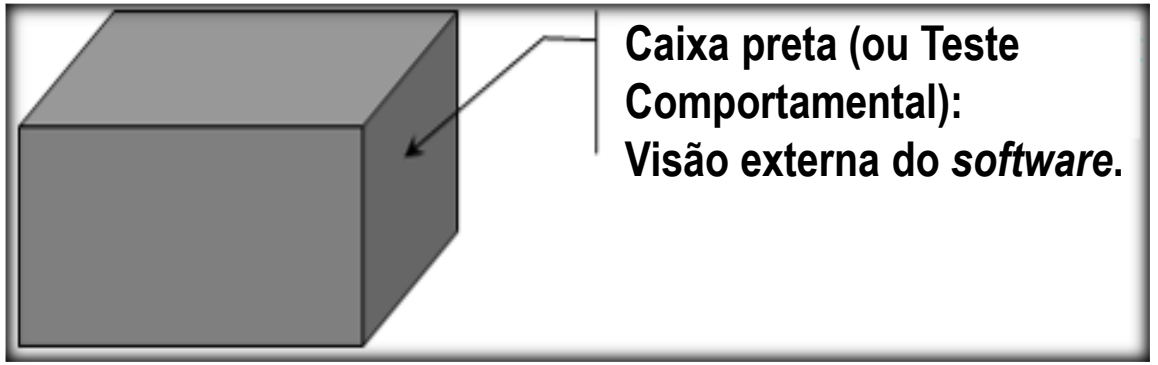
- Teste Alfa: Os usuários são levados a testar o software desde seus estágios iniciais de instalação, até a sua operação completa. Tudo realizado em um ambiente especial, controlado pelos desenvolvedores.

- Teste Beta: É acompanhado por uma versão release, é realizado exclusivamente no habitat dos usuários em ambiente descontrolado. Sem a presença do desenvolvedor, porém monitorados por este, ao contrário do teste Alfa.

Testes de software: Caixa preta e Caixa branca

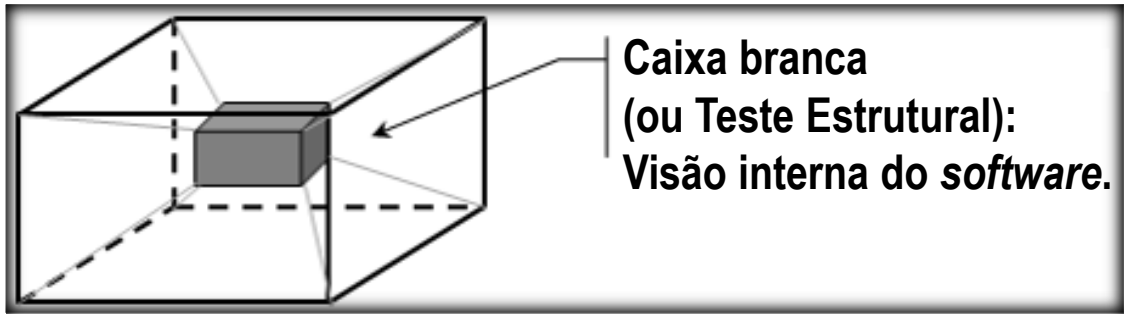
- Os testes Caixa preta e Caixa branca são guiados pelo código-fonte, métricas de software aplicáveis e se realmente estão atendendo aos requisitos com um bom desempenho e segurança.

- Teste Caixa preta (Teste Comportamental): Visa identificar as falhas em seu comportamento externo, com foco nos requisitos funcionais e conduzidos na interface de software com o usuário e com o hardware.



Fonte: adaptado de: Moreno (2023).

- O Teste Caixa branca (Teste Estrutural): É focado nos possíveis erros internos de estrutura dos componentes do sistema de software.



Fonte: adaptado de: Moreno (2023).

Testes de software: Unitário, de Integração, Aceitação e Regressão

- Teste Unitário: São realizados pelo próprio desenvolvedor durante o processo de codificação e são essenciais para identificar erros logo no início do desenvolvimento.
- Exemplo: Com base no método de teste Caixa branca, o desenvolvedor cria testes unitários para verificar se a funcionalidade está de acordo com a lógica de programação especificada no caso de teste.
- Teste de Integração: São realizados para verificar a troca de mensagens entre diferentes módulos ou componentes do software.
    - Exemplo: Em fase de teste Alfa, um analista de sistemas valida a implantação do componente de software. Para validação do software por parte do cliente, é necessário integrar o componente ao ambiente operacional e fazer os principais testes de integração, como: transferência de dados, interface de hardware e outros.
    - Teste de Regressão: São realizados para garantir que as mudanças no software, alterações feitas no código-fonte, correções de bugs ou implementação de novas funcionalidades, não tenham introduzido novos problemas.

Interatividade

A implantação do software é acompanhada por testes comportamentais, com o objetivo de identificar no software falhas e omissões em relação aos requisitos funcionais. Assinale a alternativa correspondente ao tipo de teste realizado e sua justificativa.

a) Teste alfa, porque é o primeiro teste que usuários iniciam nas operações do software.

b) Teste beta, porque se baseia em casos de uso quando o software entra em operação.

c) Teste caixa branca, porque identifica falha no comportamento interno do software.

d) Teste caixa preta, porque identifica falha no comportamento externo do software.

e) Teste de regressão, porque identifica falha nas operações do software.

Resposta

A implantação do software é acompanhada por testes comportamentais, com o objetivo de identificar no software falhas e omissões em relação aos requisitos funcionais. Assinale a alternativa correspondente ao tipo de teste realizado e sua justificativa.

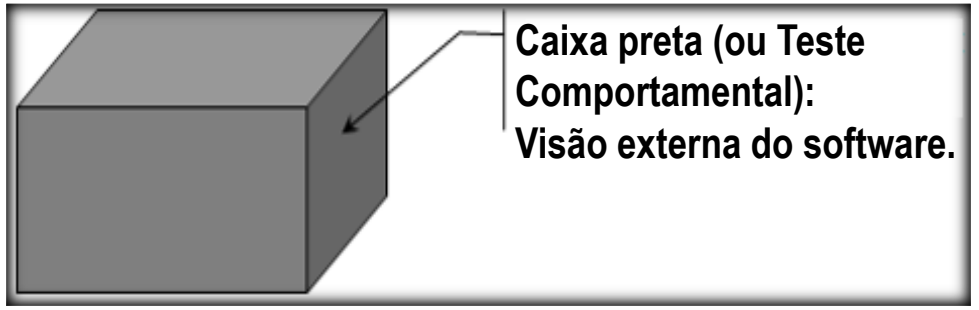
a) Teste alfa, porque é o primeiro teste que usuários iniciam nas operações do software.

b) Teste beta, porque se baseia em casos de uso quando o software entra em operação.

c) Teste caixa branca, porque identifica falha no comportamento interno do software.

d) Teste caixa preta, porque identifica falha no comportamento externo do software.

e) Teste de regressão, porque identifica falha nas operações do software.



Fonte: adaptado de: Moreno (2023).

Referências

- ARKAN, Descubra como funciona o Desenvolvimento de Softwares e seus processos. *Arkan System*. Disponível em: https://arkansystem.com.br/desenvolvimento-de-softwares_e_seus_processos/. Acesso em: 24 dez. 2020.
- FOURNIER, Roger. *Desenvolvimento e manutenção de sistemas estruturados*. São Paulo: Makron Books, 1994.
- NBR ISO/IEC 12207. *NBR ISO/IEC 12207* – Tecnologia de informação - Processos de ciclo de vida de software. Rio de Janeiro: ABNT – Ass. Brasileira de Normas Técnicas, 1998.
- PRESSMAN, R. S. *Engenharia de software*. 5. ed. Rio de Janeiro: McGraw-Hill, 2002.
- PRESSMAN, R. S. *Engenharia de software*. 7. ed. São Paulo: McGraw-Hill, 2011.
- SOMMERVILLE, Ian. *Engenharia de Software*. 9. ed. São Paulo: Pearson, 2011.
- VERSOLATTO, Fábio Rossi. *Projeto de sistemas orientado a objetos*. São Paulo: Editora Sol, 2015.

ATÉ A PRÓXIMA!