



UNIDADE II

Engenharia de *Software*

Prof. Me. Edson Moreno

Introdução

- Para ajudar as organizações desenvolvedoras de *software*, são determinadas bases fundamentais de alta qualidade e gestão eficazes de projetos.
- A primeira parte da unidade é caracterizada pela engenharia de domínio, que permite criar um conjunto de artefatos de *software*, como modelos, padrões e componentes, no qual será explorada a métrica reusabilidade, que é o principal objetivo da engenharia de *software*.
- Para aumentar a produtividade, na segunda parte da unidade, são vistos os princípios de gestão e desenvolvimento, com base em metodologias, normas e padrões da qualidade.
 - A proposta da disciplina é capacitar o aluno no conhecimento e nas práticas profissionais da Engenharia de *Software*.
 - O slide corresponde à Unidade II em resumo dos capítulos: 5. Componentização e reuso do *software*; e 6. Gestão e desenvolvimento do *software*.

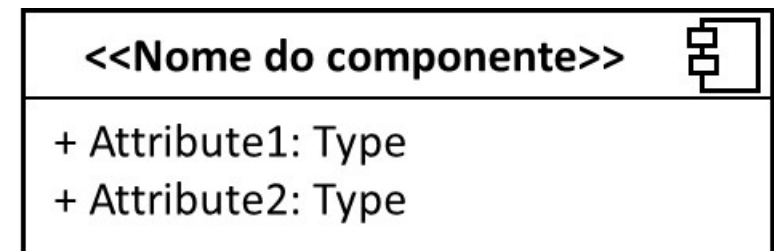
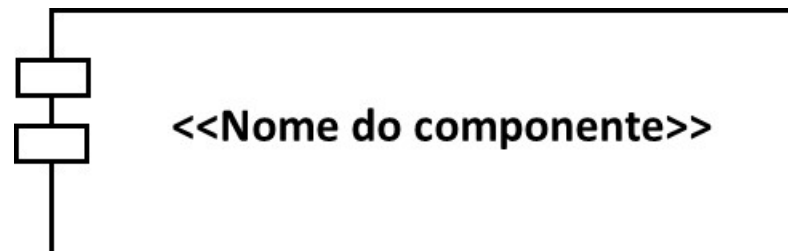
5. Componentização e reúso do *software*

- A componentização é o processo de criação de ativos digitais (módulos ou componentes), com a finalidade de reutilização em projetos de sistemas de *software*.
- A arquitetura do *software* incorpora a modularidade, isto é, o *software* é dividido em componentes nomeados separadamente e endereçáveis, frequentemente chamados de módulos, que são integrados para satisfazer os requisitos do sistema (Pressman, 2011).
- O foco nesta unidade é identificar e projetar o componente de *software*, suas formas de encapsulamento, criação dos módulos e o reúso em diversos sistemas de *software*.
 - A modularidade consiste em dividir o sistema de *software* em módulos ou componentes, que trabalham em conjunto para atingir um determinado objetivo.

Componentização: componente

- O componente de *software* tem características únicas, possível de ser implantado e substituível em um sistema, que encapsula a implementação e exhibe o conjunto de interfaces.
- O componente de *software* representa um conjunto de programas (ou classes), uma estrutura endereçável e independente com uma função específica.
- Na modelagem padrão UML, o bloco que representa o componente é desenhado com uma caixa e dois *tabs* do lado esquerdo, e o nome do componente.

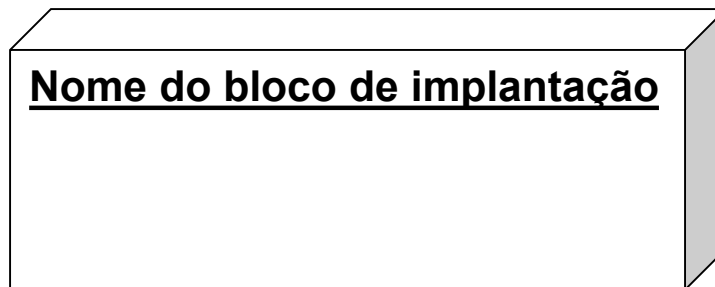
Representações de componentes de *software*:



Componentização: módulo

- O **módulo** é um atributo individual do *software* que permite gerenciar apenas um programa, um *software* ou um sistema.
- A **modularidade** do *software* visa a uma interpretação simples do projeto, que permite boas análises para o suporte e a manutenção do sistema.
- Na modelagem padrão UML, o bloco que representa o módulo é chamado de bloco de implantação (*deployment*), chamado também de bloco de distribuição. O bloco de implantação é desenhado com uma caixa com faixas de sombreados no topo e à direita, dando a noção de 3D.

Representações do bloco de implantação:



Fonte: adaptado de: Moreno (2023).

“Nós” de processamento

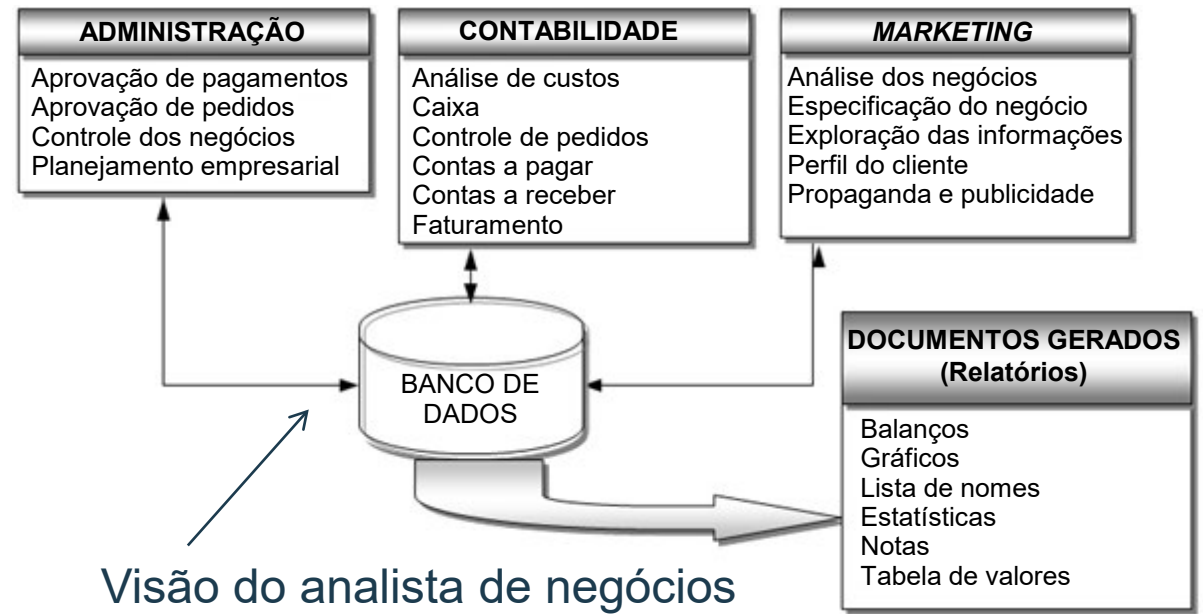
- O diagrama de implantação, ou de distribuição, mostra como os componentes são configurados para execução, em “nós” de processamento (Larman, 2007).
- Um “nó” de processamento é um recurso computacional que permite a execução de um sistema de *software*, ou parte dele, como um componente.
- O “nó” pode ser um computador, um dispositivo móvel, uma estrutura de memória ou mesmo um dispositivo periférico.
- Um “nó” precisa ter um endereço, que é o identificador de um módulo ou componente de *software*.

Saiba mais

IBM. *Nós de Processamento de Mensagem Definidos pelo Usuário*. IBM® Integration Bus, 24 ago. 2022. Disponível em: <https://www.ibm.com/docs/pt-br/integration-bus/10.0?topic=nodes-user-defined-message-processing>. Acesso em: 16 jan. 2024.

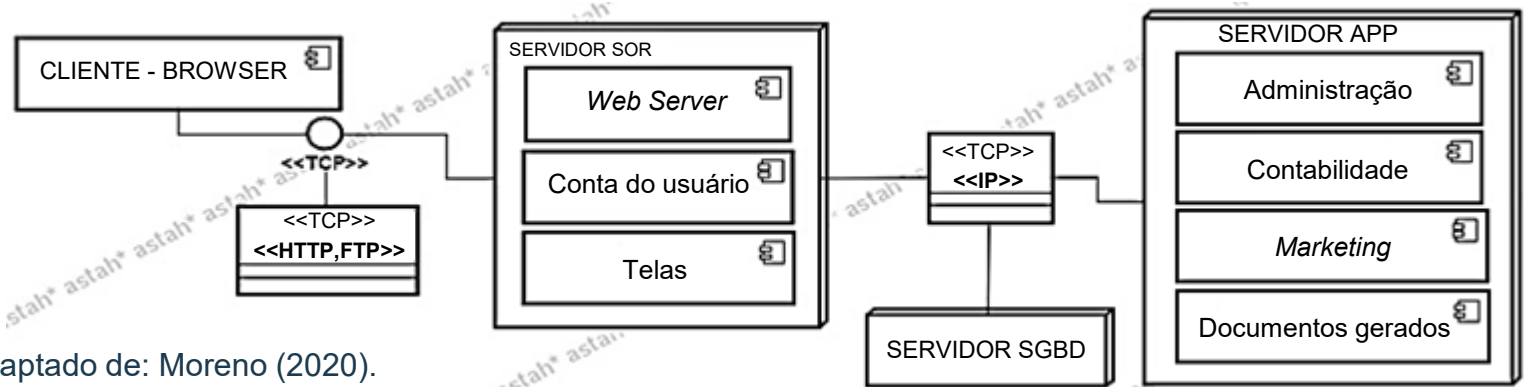
Análise de ERP I: identificação e construção de módulos e componentes

- A figura de cima representa um modelo de Requisito do Usuário (RU), desenhado por um analista de negócios.
→ Modelagem do negócio.
- A figura debaixo representa um modelo de Requisito do Sistema (RS), desenhado por um engenheiro de *software*.
→ Modelagem do sistema de *software*.



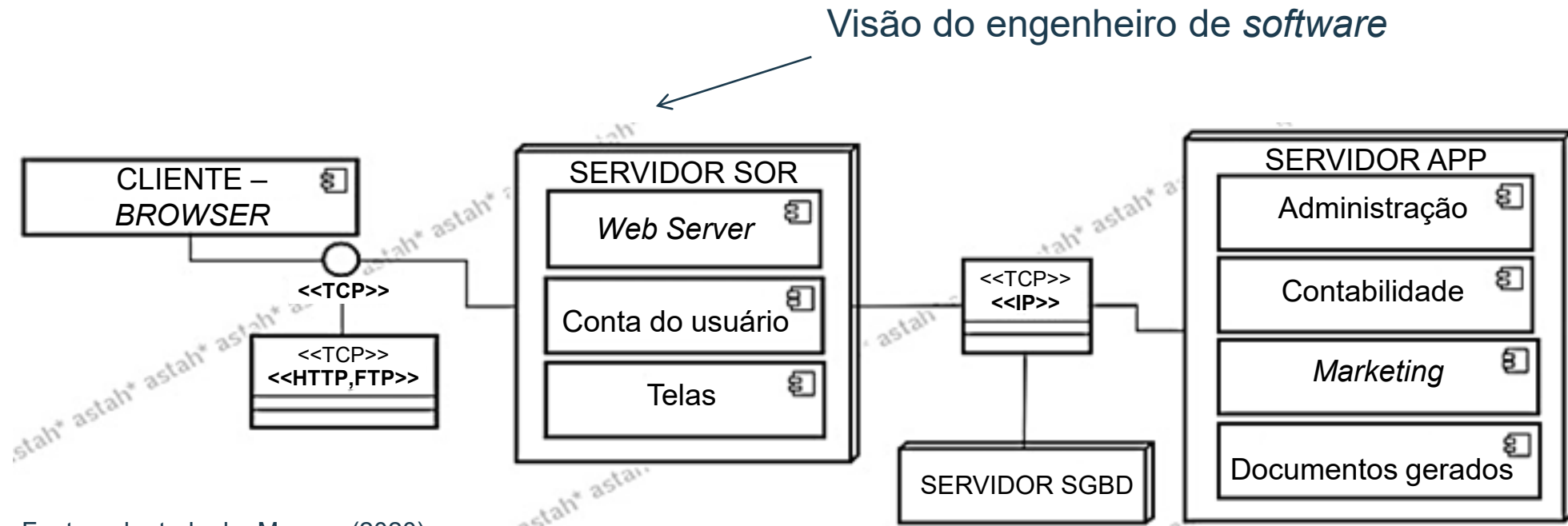
Visão do analista de negócios

Visão do engenheiro de *software*



Fonte: adaptado de: Moreno (2020).

Análise de ERP II: interpretação do modelo de sistema

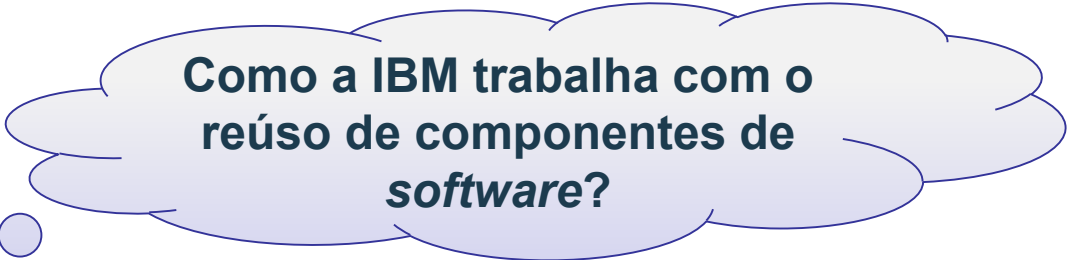


Fonte: adaptado de: Moreno (2020).

- Observe os estereótipos **<<TCP>>** que indicam conexões com endereçamento próprio (IP), logo, cada um se refere a um “nó” no sistema de *software*.
- Nos “nós”, estarão: como componente “CLIENTE – BROWSER” e como módulos “SERVIDOR SOR”, “SERVIDOR APP” e “SERVIDOR SGBD”.

Engenharia de domínio e reusabilidade do *software*

- A reusabilidade do *software* é uma métrica de qualidade usada para avaliar quanto um programa ou parte dele pode ser usada em outras aplicações.
- À medida que são produzidos componentes de *software*, esses podem ser combinados de forma lógica e racional para produzir outros novos componentes, módulos ou mesmo novos produtos de *software*.
- A prática da diversificação é adquirir e construir um repertório de alternativas, a matéria-prima do projeto: componentes, soluções de componentes e conhecimento.



Como a IBM trabalha com o reúso de componentes de *software*?

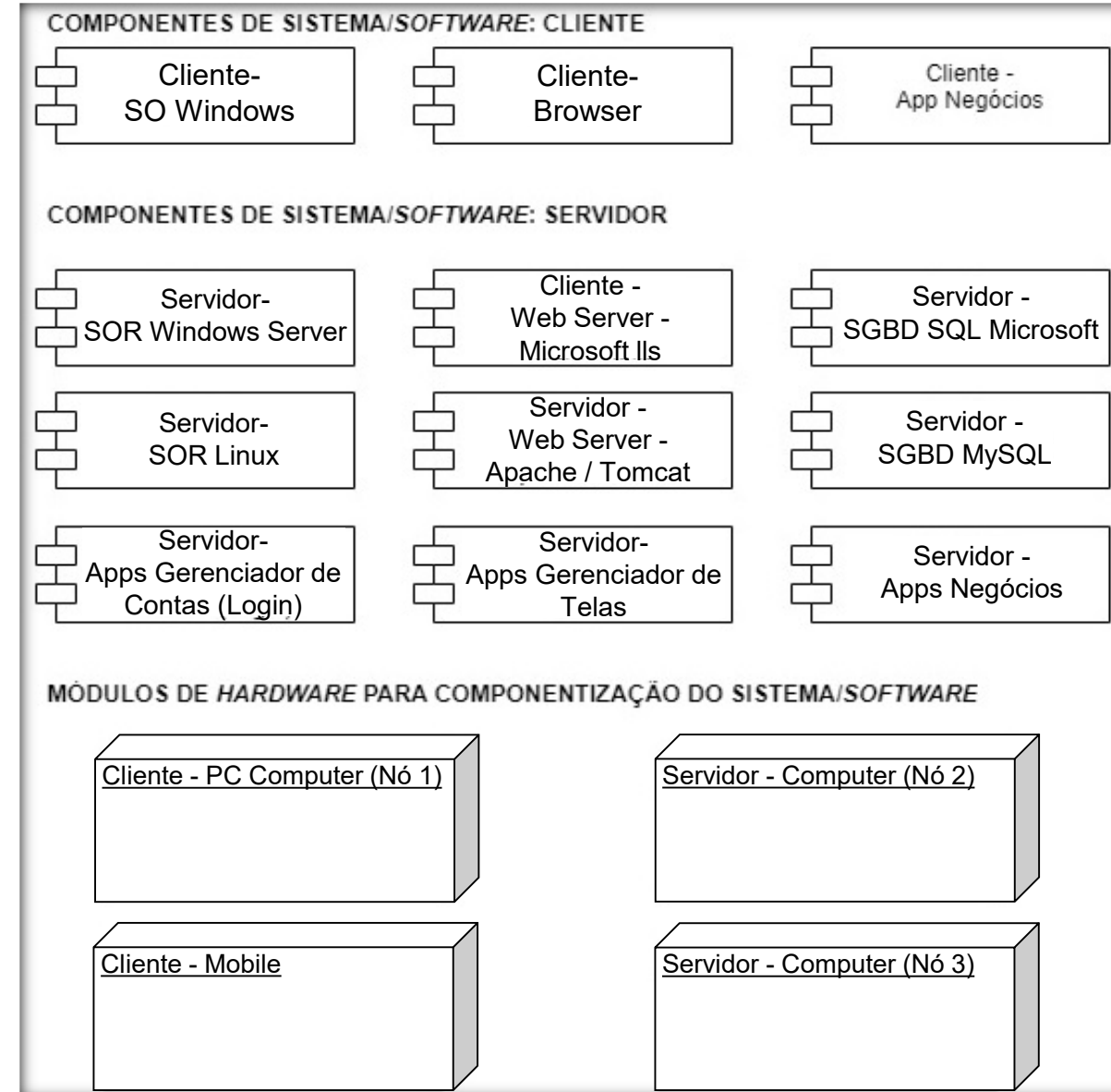
IBM®. *Componentes*. Rational Software Architect Standard Edition, 05 mar. 2021. Disponível em: <https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=diagrams-components>. Acesso em: 17 jan. 2024.

Diversificação: repertório de componentes

- A diversificação fornece diversos componentes e módulos, disponíveis para o projetista.
- Essa prática flexibiliza e aumenta a produtividade do projeto de sistemas de *software*.
- Na figura, diversos componentes e módulos são criados e passam a fazer parte do repertório de alternativas que servem para a construção de sistemas de *software*.



Fonte: Clipart.



Fonte: adaptado de: Moreno (2023).

Interatividade

O componente é um bloco modular do *software*. Em relação ao componente de *software*, analise se as afirmativas abaixo são verdadeiras ou falsas.

- I. Mostra uma determinada sequência de operações do sistema de *software*.
- II. Possui uma estrutura endereçável e independente que representa uma função específica.
- III. Tem características únicas possíveis de ser implementado ou substituído.

Assinale a alternativa correta quanto à análise:

- a) I, II e III são falsas.
- b) I, II e III são verdadeiras.
- c) I e II são verdadeiras.
- d) I e III são verdadeiras.
- e) II e III são verdadeiras.

Resposta

O componente é um bloco modular do *software*. Em relação ao componente de *software*, analise se as afirmativas abaixo são verdadeiras ou falsas.

- I. Mostra uma determinada sequência de operações do sistema de *software*.
- II. Possui uma estrutura endereçável e independente que representa uma função específica.
- III. Tem características únicas possíveis de ser implementado ou substituído.

Assinale a alternativa correta quanto à análise:

- a) I, II e III são falsas.
- b) I, II e III são verdadeiras.
- c) I e II são verdadeiras.
- d) I e III são verdadeiras.
- e) II e III são verdadeiras.

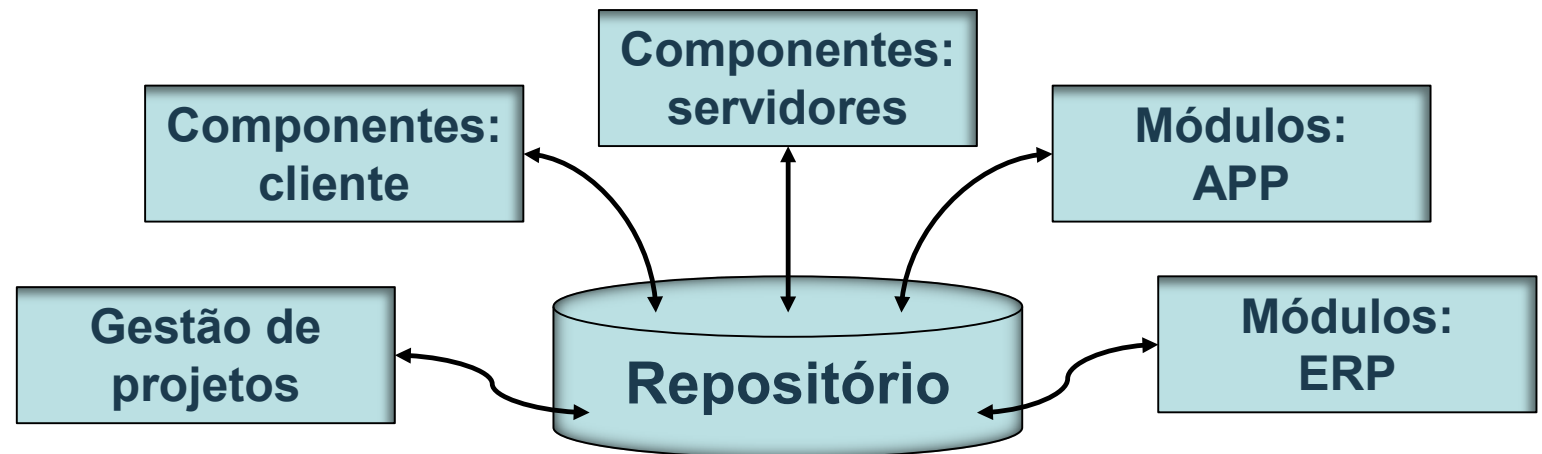
Engenharia de domínio: repositório do desenvolvimento de *software*

- No domínio da aplicação, a engenharia de domínio tem como objetivos: identificar, construir, catalogar e disseminar um conjunto de componentes de *software* que tenham aplicabilidade para o *software* já existente e futuro.



Produção de peças (componentes) de *software*.

- O domínio da aplicação é como uma família de produtos – aplicações com funcionalidades específicas com objetivo de partilhar esses componentes em sistemas de *software*.
- Disponível ao projetista, abaixo é apresentado o modelo de acesso ao repositório (repertório de alternativas para o projeto de *software*).

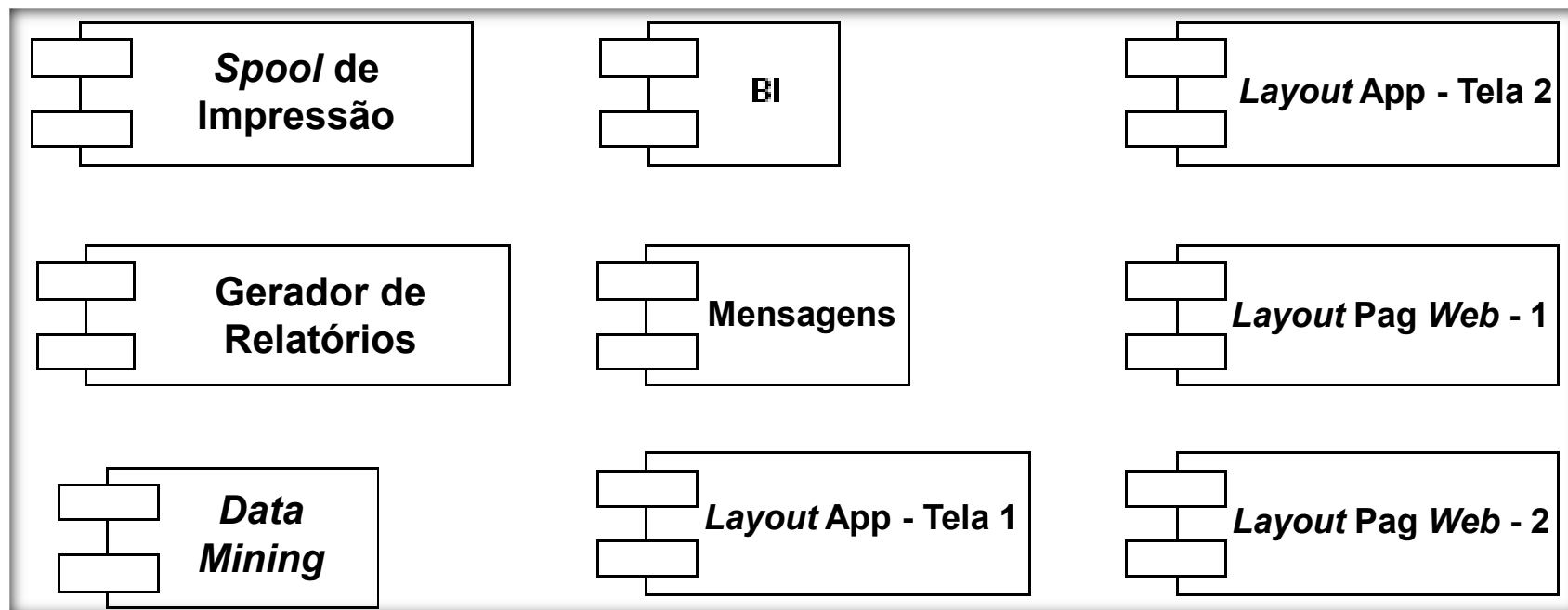


Análise: modelo de repositório para desenvolvimento de aplicação

Observe o modelo abaixo com um repertório de alternativas para desenvolver aplicações para o sistema BI (*Business Intelligence*):

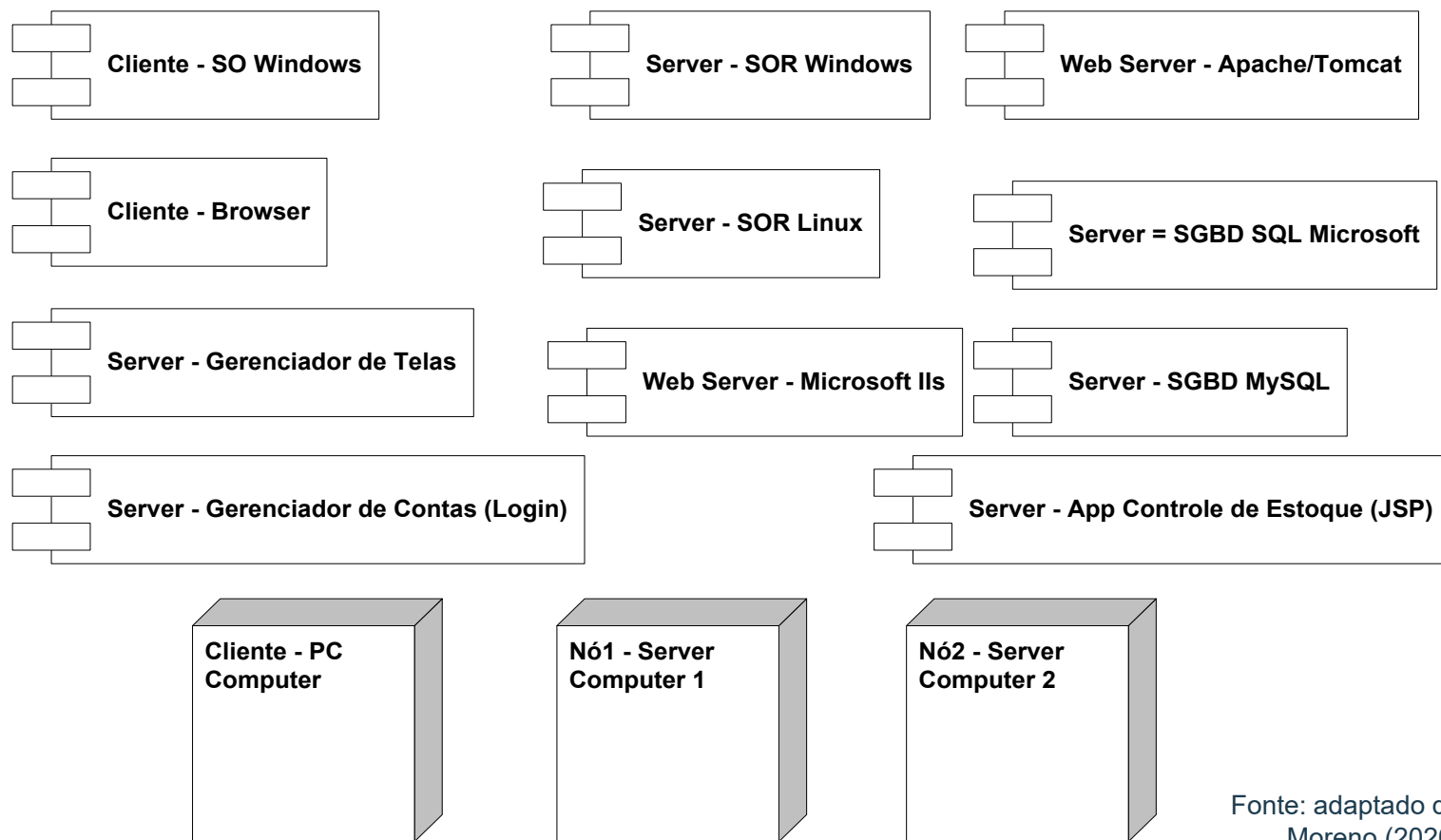
- Esse repertório de alternativas está armazenado em um repositório.
- No conjunto, vemos os blocos funcionais (componentes): 1. Aplicação BI; 2. *Spool* de impressão; 3. Bloco funcional *Data Mining* (suporte para o BI); 4. Blocos funcionais para geradores de relatórios e mensagens; 5. *Layouts* de telas para app e web.

Fonte: adaptado de: Moreno (2023).



Componentes/Implantação I: seleção de componentes

- A modelagem de construção com base no diagrama de componentes/implantação é a prática da convergência, em que o projetista escolhe os elementos do repertório que satisfaça os requisitos do sistema (RS) e os resume em uma particular configuração de componentes encapsulada para a criação do produto final.
- Ao lado, vemos alguns dos componentes disponíveis para o projeto do sistema de *software*.



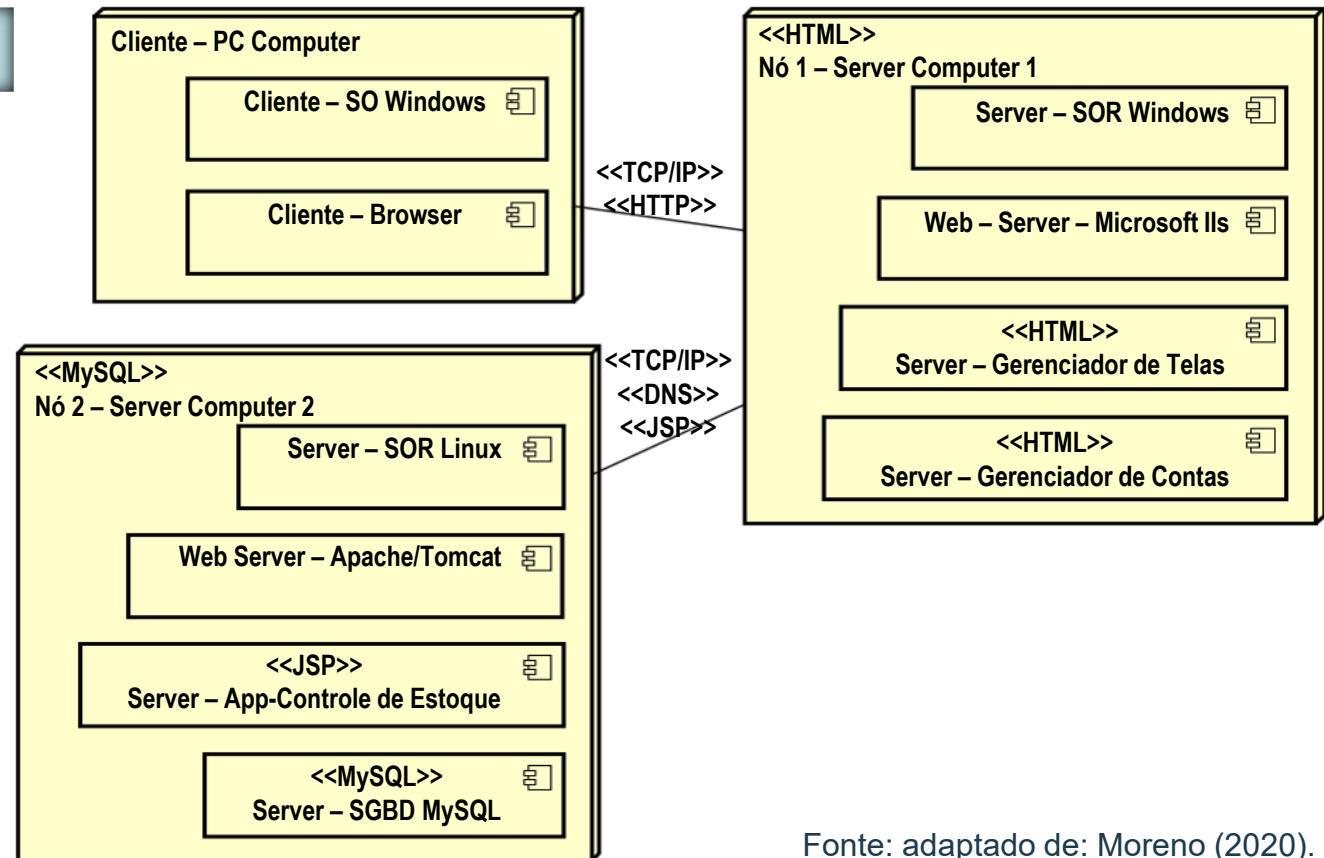
Componentes/Implantação II: convergência e módulos de implantação

- A convergência dos componentes em módulos de implantação, bem como suas ligações permitem criar o projeto de infraestrutura do sistema de *software*.

Diversificação

Convergência

- Ao lado, vemos o modelo de arquitetura do sistema, que é o resultado da lógica de convergência de componentes para a construção de um sistema de *software*.



Fonte: adaptado de: Moreno (2020).

Projeto de componentes

Os cinco princípios básicos de um componente, segundo Cheesman e Daniels (2000), são:

- Um componente obrigatoriamente deve possuir uma especificação.
- Um componente obrigatoriamente deve possuir uma implementação.
- Um componente obrigatoriamente deve seguir uma padronização.
- Um componente obrigatoriamente deve ter a capacidade de ser empacotado em módulos.
- Um componente obrigatoriamente deve ter a capacidade de ser distribuído.



Vale a pena lembrar que:

- ➔ O componente é a parte modular, possível de ser implantada e substituível de um sistema, que encapsula a implementação e exhibe o conjunto de interfaces.
- ➔ A componentização é o processo de criação de ativos digitais com a finalidade de reutilização em projetos de sistemas de *software*.

Diagrama de componentes

- O diagrama de componentes está associado a todos os requisitos do sistema (RS), pode estar associado à linguagem de programação, à topologia e protocolos de rede, ao SGBD, bem como interfaces de entradas e saídas.
 - ➔ Todos os recursos do sistema que dão apoio ao *software* devem estar contemplados no diagrama de componentes.
- Cada componente pode representar módulos de código-fonte, bibliotecas, formulários, arquivos de ajuda, módulos executáveis e outros.

Fonte: adaptado de: Moreno (2024).

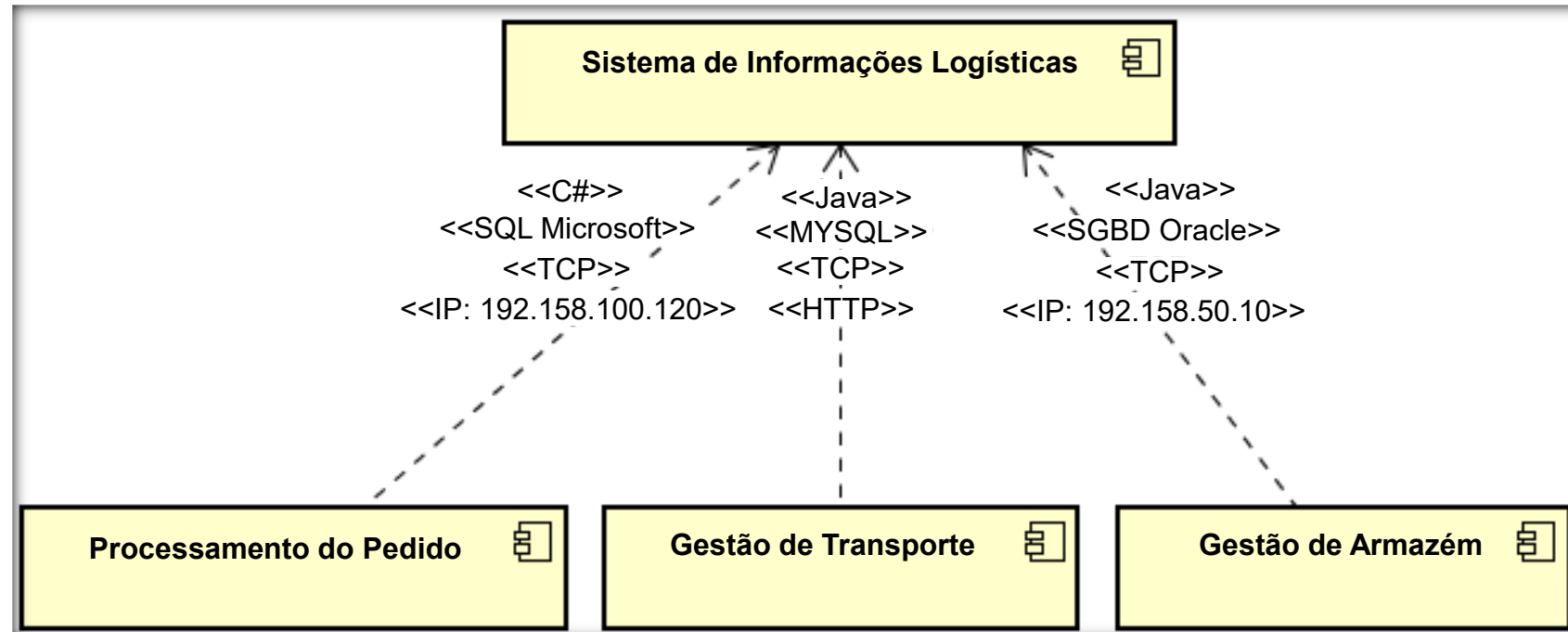


Diagrama de implantação

- O diagrama de implantação pode ser considerado uma associação de diversos componentes que determinam as necessidades de *hardware*, da estruturação dos serviços de dados e característica da rede de computadores do sistema de *software*.
- O diagrama de implantação é usado também para integrar outros subsistemas, associando diversos módulos, como é mostrado no modelo de arquitetura abaixo.
- Ao lado, vemos um modelo de arquitetura do sistema para uma aplicação em caixa eletrônico.

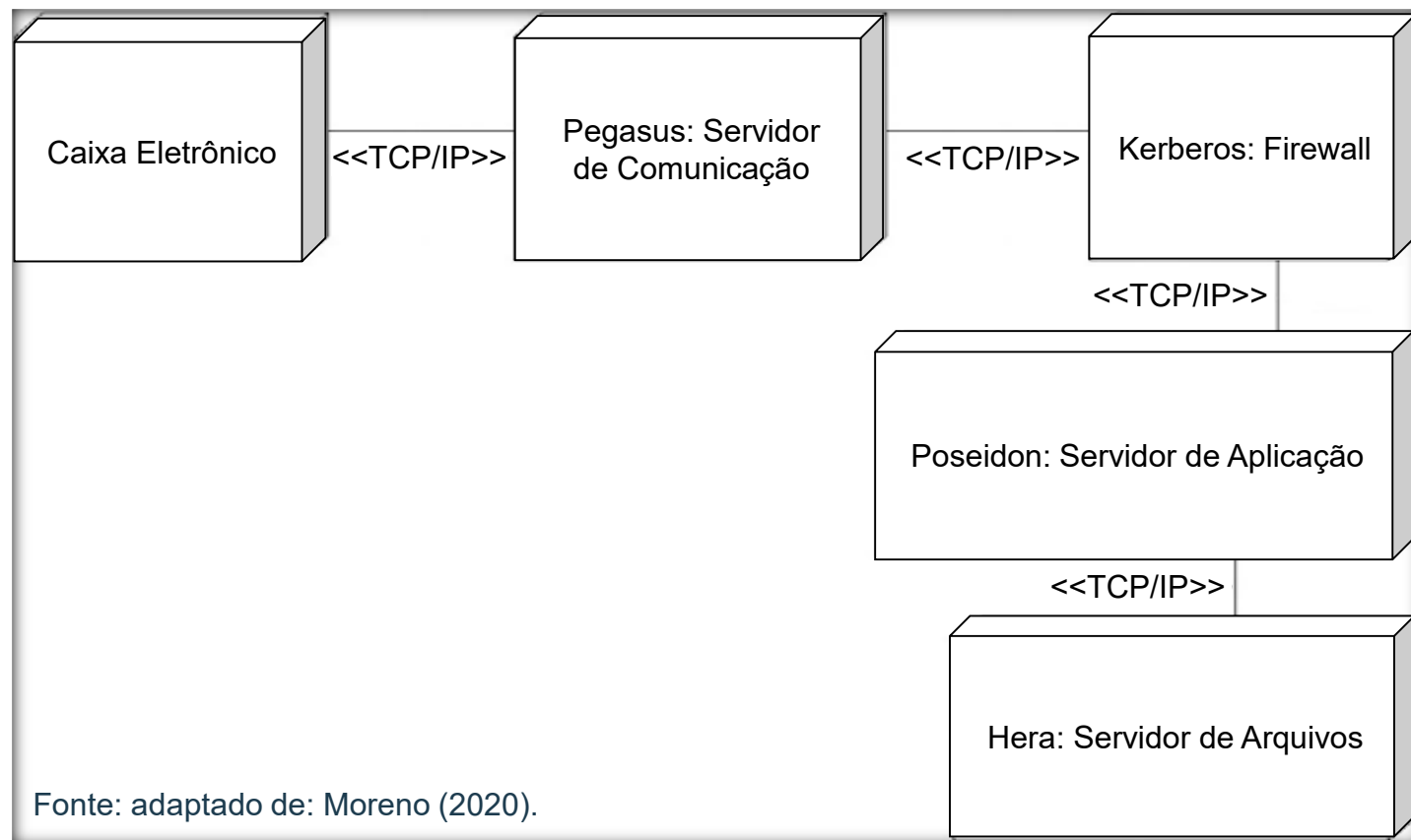
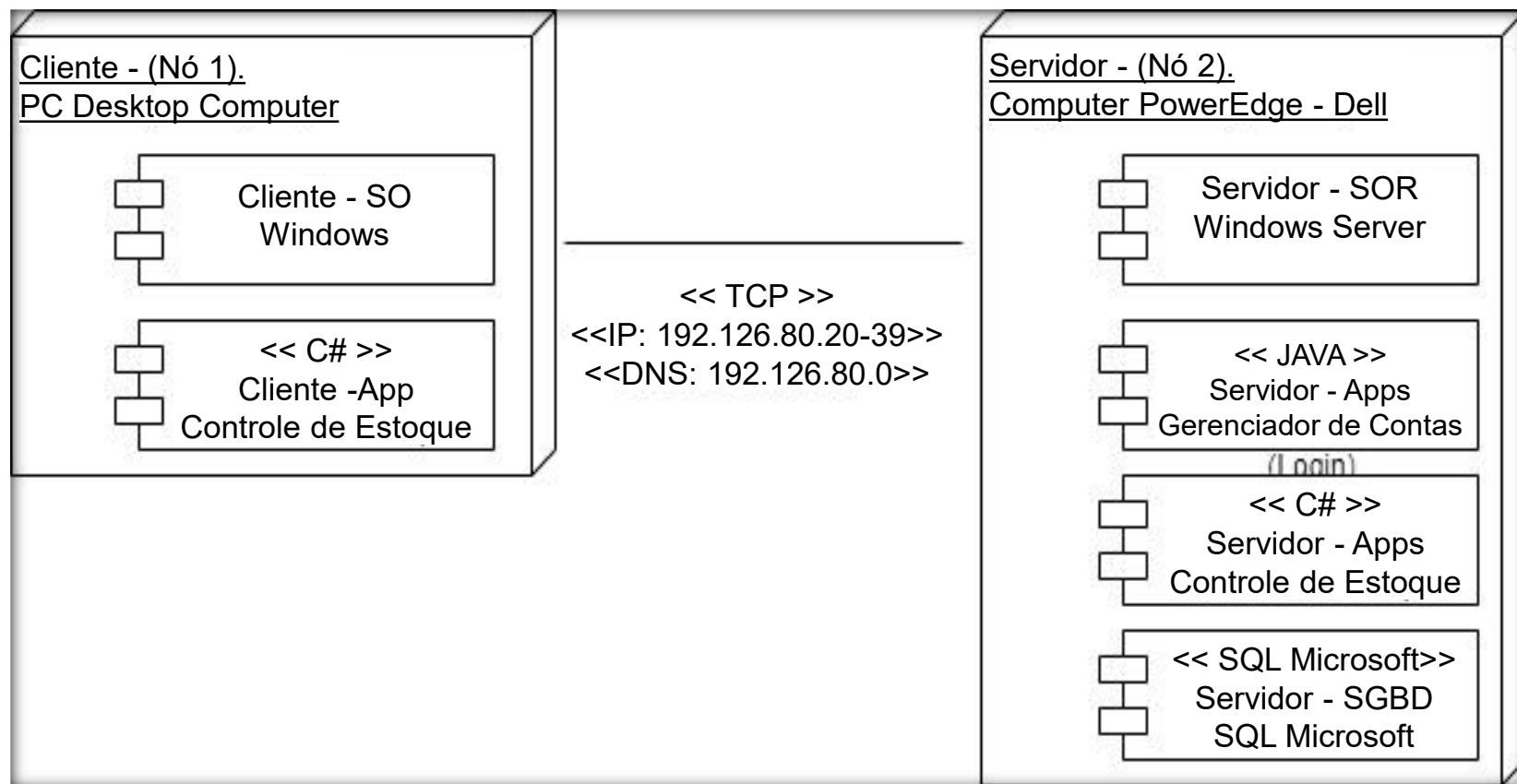


Diagrama de implantação/componentes

Em uma visão mais ampla e detalhada, vemos no modelo a integração entre os blocos de componentes, blocos de implantação e respectivos estereótipos.

- Na análise vemos neste modelo dois “nós”, que indicam dois módulos de sistemas. Um do cliente e o outro do servidor.
- Vemos também as tecnologias implementadas para suas ligações.



Interatividade

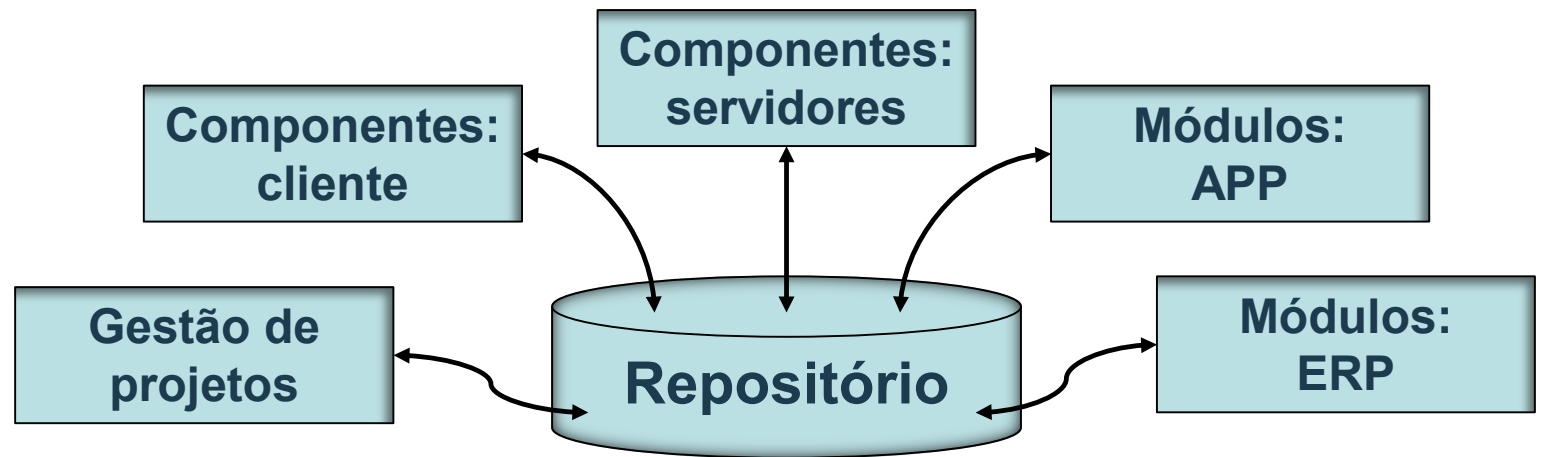
“Este elemento é utilizado como meio de armazenamento, gestão e compartilhamento de objetos, componentes, modelos, documentos ou quaisquer outros artefatos de *software* ou do sistema produzidos em um ambiente de desenvolvimento”. Qual o nome desse elemento?

- a) Banco de conhecimentos.
- b) Banco de dados.
- c) Elemento do processo de negócio.
- d) *Feedback*.
- e) Repositório.

Resposta

“Este elemento é utilizado como meio de armazenamento, gestão e compartilhamento de objetos, componentes, modelos, documentos ou quaisquer outros artefatos de *software* ou do sistema produzidos em um ambiente de desenvolvimento”. Qual o nome desse elemento?

- a) Banco de conhecimentos.
- b) Banco de dados.
- c) Elemento do processo de negócio.
- d) *Feedback*.
- e) **Repositório.**



6. Gestão e desenvolvimento do *software*

Esta seção se refere aos princípios de gestão e desenvolvimento, com base em metodologias, normas e padrões da qualidade.

- Em relação a projetos, será abordado o Guia PMBOK® – *Project Management Body of Knowledge* (Guia do Conhecimento em Gerenciamento de Projetos), que descreve bons processos e práticas de gestão de projetos.
- Destaque para as práticas do desenvolvimento de *software* com as metodologias ágeis.

Modelos de organização para o desenvolvimento de *software*, com estruturas de padrões e modelos de qualidade, tais como:

- SPICE – ISO 15504 (Processo de Avaliação de Capacidade de Processos de *Software*).
- ISO 9000 (Sistemas de Gestão da Qualidade).
- CMMI – *Capability Maturity Model Integration* (Modelo de Maturidade em Capacitação – Integração).

Fundamentos da gestão de projetos e o guia PMBOK®

- Um projeto do sistema de *software* fornece detalhes sobre as estruturas de dados, arquitetura, interfaces e componentes necessários para implementar o sistema.

O modelo atual de Gestão de Projetos tem origens no desenvolvimento de sistemas. Esse modelo é amplamente usado pela maioria das empresas em todos os ramos de negócio do mundo. Esse modelo é chamado de:

PMBOK® – *Project Management Body of Knowledge*
(Guia do Conhecimento em Gerenciamento de Projetos).

- O Guia PMBOK® é publicado e distribuído pelo PMI – *Project Management Institute*, que é uma norma reconhecida para a profissão de gerenciamento de projeto.

Gerenciamento de projetos com PMBOK®

- O bom projeto viabiliza o uso do sistema por um longo período de tempo, dispondo de recursos que permitam adaptá-lo a novas mudanças, correções e integração a novos ambientes operacionais.
- O Guia PMBOK® é uma base sobre a qual as organizações podem criar metodologias, políticas, procedimentos, regras, ferramentas, técnicas e fases do ciclo de vida necessários para a prática do gerenciamento de projetos (PMBOK, 2017).

Saiba mais

Fonte: ÁVILA, Márcio. *PMBOK e Gerenciamento de Projetos*. Disponível em: <http://www.mhavila.com.br/topicos/gestao/pmbok.html>. Acesso em: 18 jan. 2024.

© 2006-2015, Márcio d'Ávila



Componentes-chave do guia PMBOK®

- O ciclo de vida do projeto pelo PMBOK® percorre as fases: início do projeto, organização e preparação, execução do trabalho e terminar o projeto.
- O gerenciamento de projetos permite ter uma abordagem estruturada e sistemática para planejar, executar e controlar projetos de forma eficiente.

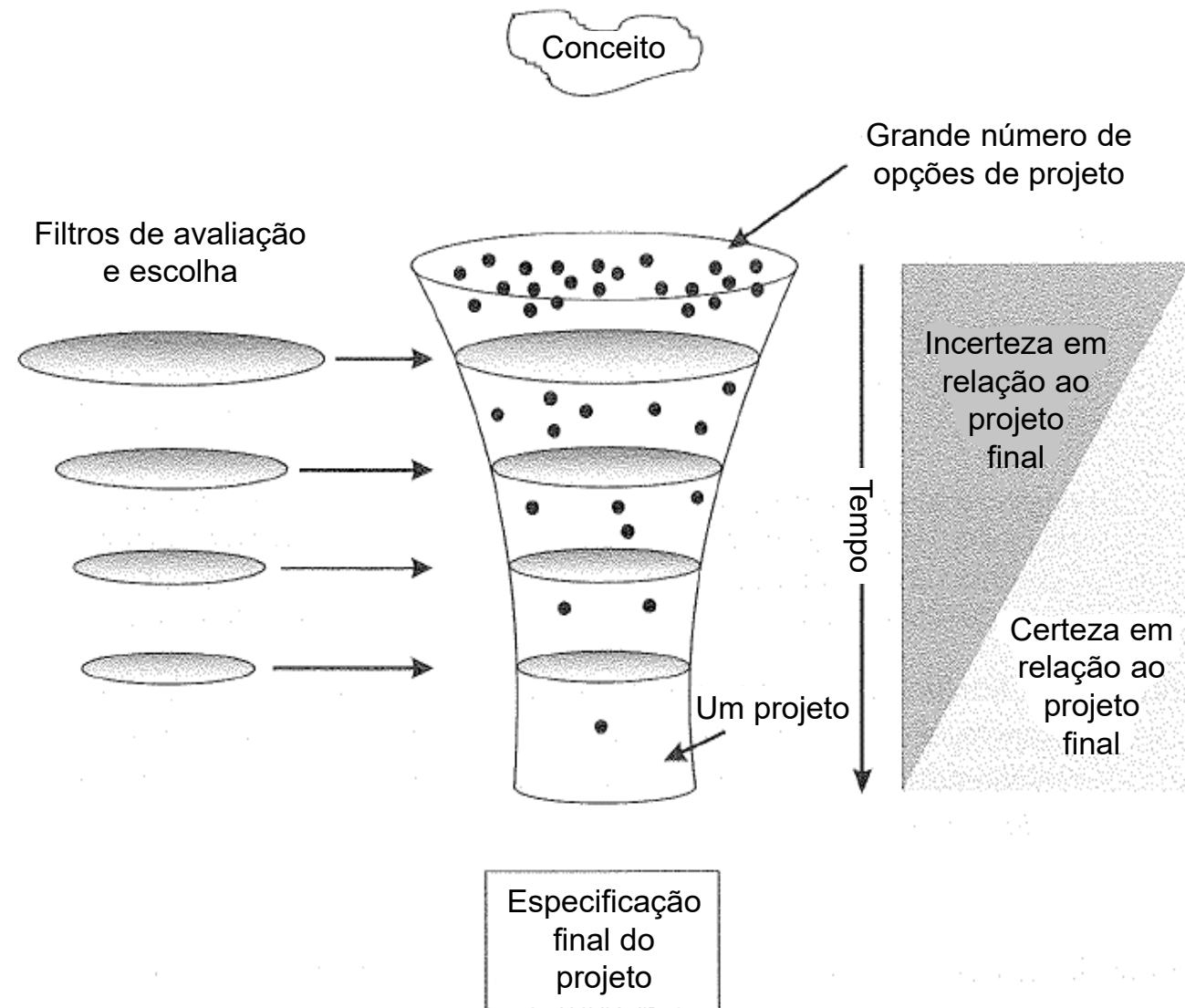
A estrutura do Guia PMBOK® possui diversos componentes-chave que são mostrados no quadro abaixo:

Componentes-chave do Guia PMBOK®
Ciclo de vida do projeto
Fase do projeto
Revisão de fase
Grupo de processos de gerenciamento de projetos
Processo de gerenciamento de projetos
Área de conhecimento em gerenciamento de projetos

O projeto

No projeto, são elaborados vários processos que monitoram todas as fases do projeto.

- Para cada processo, existem várias alternativas de componentes, métodos de montagem e de várias ferramentas.
- Todo esse aparato deve ser detalhado e estudado.
- E se reduz progressivamente o número de alternativas até o projeto final.



Áreas de conhecimento e gerenciamento de projetos

São basicamente dez as áreas de conhecimento que identificam as principais gerências que atuam no projeto de acordo com o PMBOK® (2017).

As áreas de conhecimento são mostradas abaixo:

- Cada área de conhecimento (ou gerência) é associada a um ou mais processos do grupo de processos, de influência no projeto.

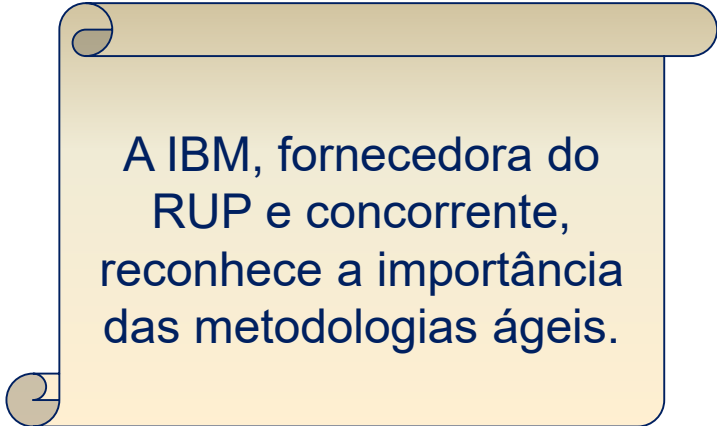


Metodologias ágeis

Com emprego de técnicas consideradas eficientes no desenvolvimento do *software*, essas técnicas passaram a se chamar de metodologias ágeis.

Nas metodologias ágeis, serão abordados os seguintes itens:

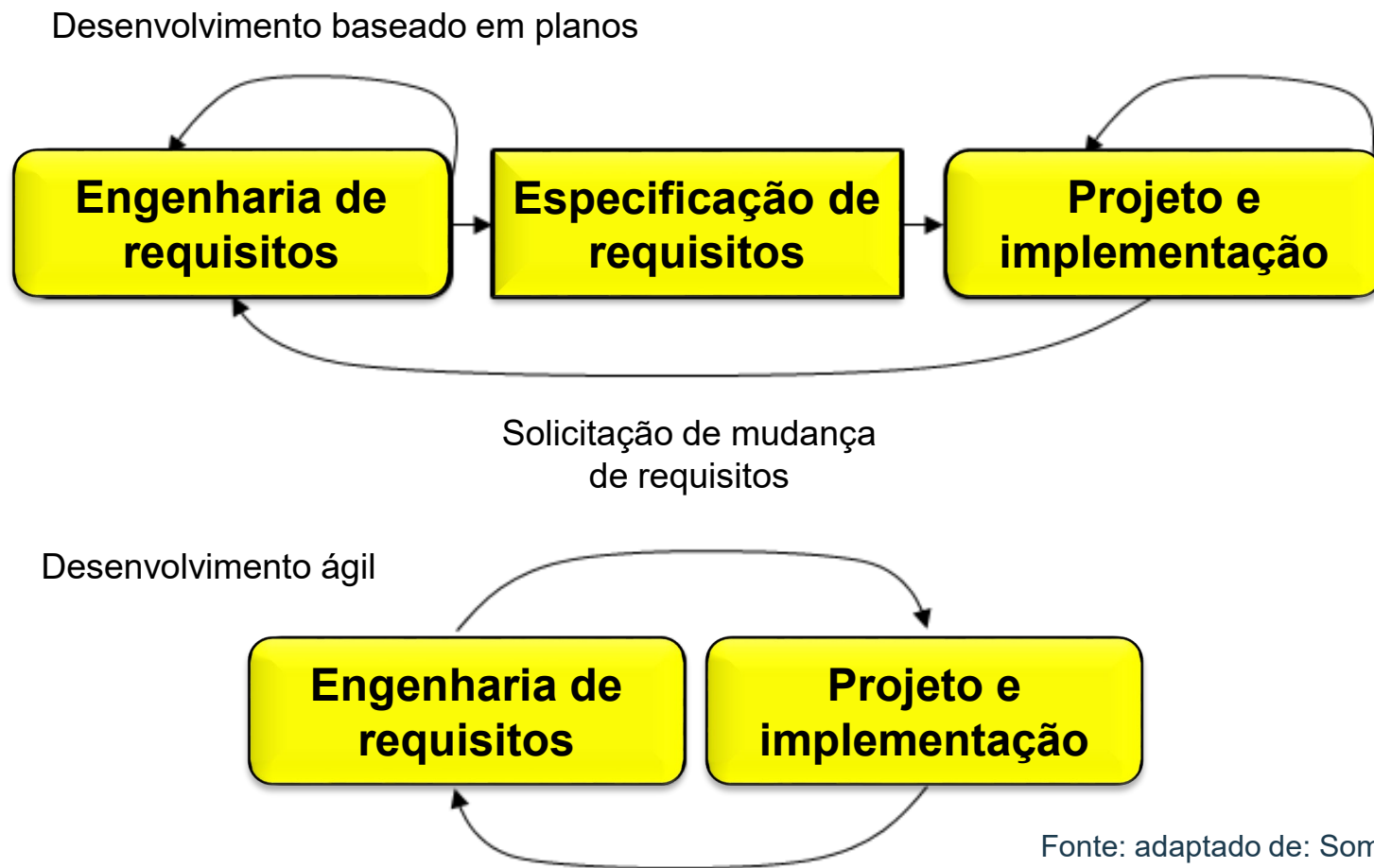
- Manifesto para desenvolvimento ágil de *software*.
- Metodologias ágeis: XP, SCRUM, FDD, DSDM, Crystal e AM.



A IBM, fornecedora do RUP e concorrente, reconhece a importância das metodologias ágeis.

Desenvolvimento estruturado *versus* metodologias ágeis

- Observe a figura: o desenvolvimento prescritivo (desenvolvimento baseado em planos) é caracterizado por seguir uma abordagem mais estruturada e sequencial, com etapas bem definidas, especificadas e detalhadas antes do início do projeto, Geralmente, utilizando o modelo de processo cascata.



Manifesto para desenvolvimento ágil de *software* I

Por meio do *Manifesto for Agile Software Development* (*Manifesto para Desenvolvimento Ágil de Software*), publicado nos dias 11 a 13 de fevereiro de 2001, por Kent Beck e mais 16 notáveis desenvolvedores, que se reuniram para defender as seguintes regras:

Fonte: adaptado de: BECK, Kent. *et al. Manifesto para Desenvolvimento Ágil de Software*, 2001.

Disponível em:

<https://agilemanifesto.org/iso/ptbr/manifesto.html>.

Acesso em: 30 jun. 2020.

Manifesto para Desenvolvimento Ágil de *Software*

Estamos descobrindo maneiras melhores de desenvolver *software*, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Por meio desse trabalho, passamos a valorizar:

Indivíduos e interações mais que processos e ferramentas.

Software em funcionamento mais do que documentação abrangente.

Colaboração com o cliente mais do que negociação de contratos.

Responder a mudanças mais que seguir um plano.

Manifesto para desenvolvimento ágil de *software* II

A frase que finaliza esse manifesto é: “Mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda” (Beck, 2001). Pode ser explicada com base na figura abaixo.

- Enquanto o RUP (lado direito da figura) é extremamente rígido com altos níveis de controle e forte documentação, as metodologias ágeis (no centro e lado esquerdo da figura) caminham ao contrário e, mesmo assim, as metodologias ágeis não infringem uma sólida prática da Engenharia de *Software*.



Manifesto para desenvolvimento ágil de *software* – princípios

Princípios	Descrição
Envolvimento do cliente	O cliente fornece e prioriza novos requisitos.
Entrega incremental	Os requisitos são incluídos em incrementos sucessivos.
Pessoas, não processos	Habilidades e comunicação informal são exploradas.
Aceitar as mudanças	Os requisitos sempre mudam.
Manter a simplicidade	Sem complexidade no processo de <i>software</i> .

Fonte: Sommerville (2011).



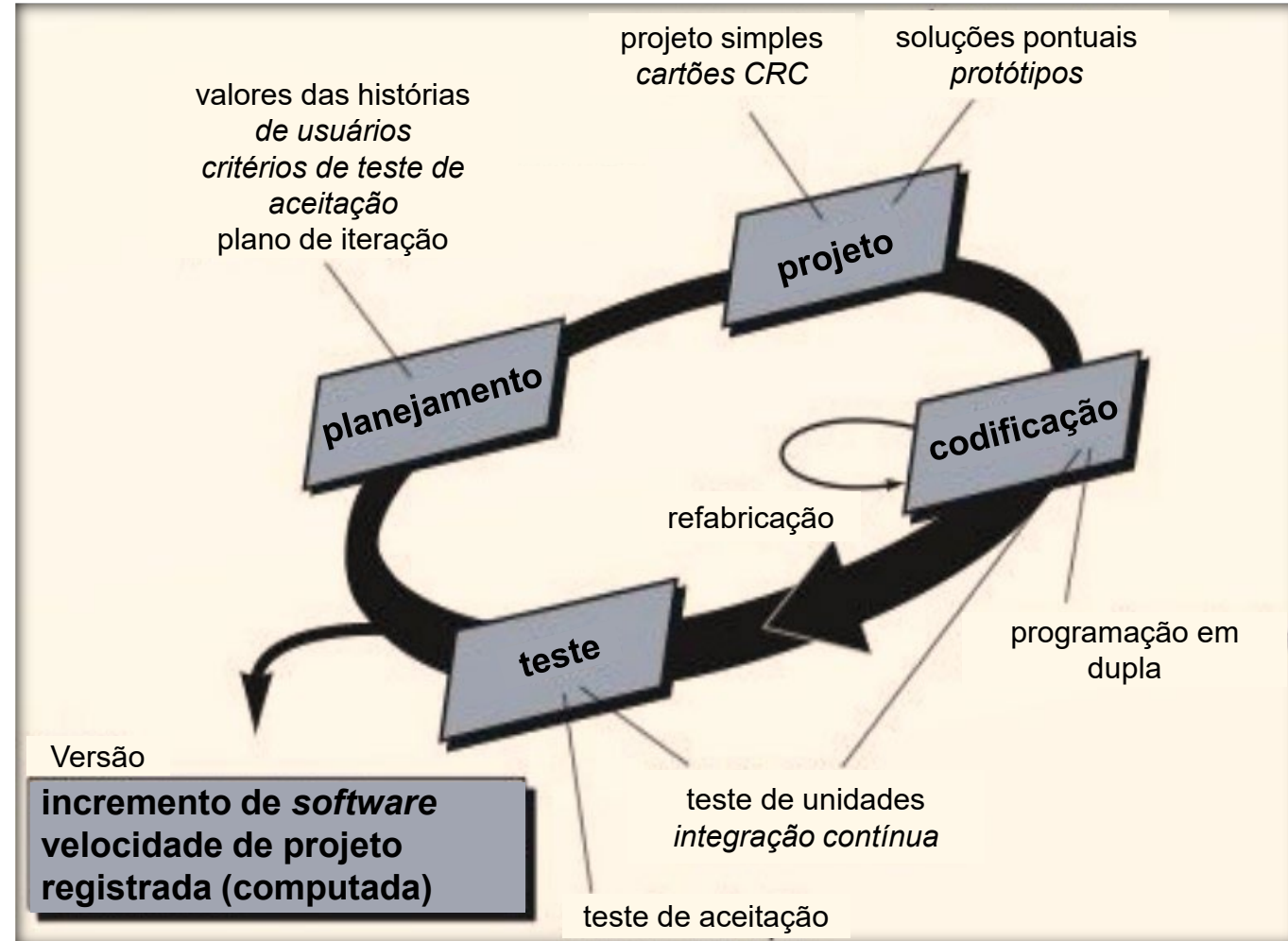
Fonte: Clipart.

Principais metodologias ágeis

- XP – *Extreme Programming* (Programação Extrema): aplicada em equipes pequenas de programação.
- SCRUM: equipes de desenvolvimento com foco em construir e integrar *software* em ambientes complexos, requisitos pouco estáveis, desconhecidos ou que mudam com frequência.
- FDD – *Feature Driven Development* (Desenvolvimento Dirigido a Funcionalidades): apropriado para implementação de funcionalidades em projetos de médio e grande porte.
- DSDM – *Dynamic Systems Development Method* (Método Dinâmico de Desenvolvimento de Sistemas): aplicado na especificação, integração e testes de componentes.
 - Crystal: conjunto de metodologias aplicadas a pequenos projetos.
 - AM – *Agile Modeling* (Modelagem Ágil): prática para modelagem e documentação do *software*.

Metodologia ágil: XP – *extreme programing* (programação extrema)

- Abordagem desenvolvida considerando boas práticas com desenvolvimento iterativo, em níveis “extremos”.
- A comunicação na XP enfatiza a colaboração estreita entre clientes e desenvolvedores, com nomenclatura única e *feedbacks* contínuos.
- São projetadas apenas as necessidades imediatas, com projetos simples e de fácil implementação de código.



Interatividade

Analise os princípios abaixo e assinale como verdadeiros ou falsos aqueles que correspondem às metodologias ágeis, e assinale a alternativa correspondente à resposta completa.

- I. Entrega de funcionalidades de *software* estruturado em períodos curtos.
- II. Envolvimento do cliente para fornecer e priorizar novos requisitos.
- III. O cliente determina os incrementos do desenvolvimento de *software*.
- IV. O projeto deve acomodar as mudanças de requisitos durante o ciclo de desenvolvimento.
- V. Os processos de *software* prescritivos são bem empregados nas metodologias ágeis.

- a) I é verdadeira.
- b) I e IV são verdadeiras.
- c) I, II, III, IV e V são verdadeiras.
- d) II, III e IV são verdadeiras.
- e) IV é verdadeira.

Resposta

Analise os princípios abaixo e assinale como verdadeiros ou falsos aqueles que correspondem às metodologias ágeis, e assinale a alternativa correspondente à resposta completa.

- I. Entrega de funcionalidades de *software* estruturado em períodos curtos.
- II. Envolvimento do cliente para fornecer e priorizar novos requisitos.
- III. O cliente determina os incrementos do desenvolvimento de *software*.
- IV. O projeto deve acomodar as mudanças de requisitos durante o ciclo de desenvolvimento.
- V. Os processos de *software* prescritivos são bem empregados nas metodologias ágeis.

- a) I é verdadeira.
- b) I e IV são verdadeiras.
- c) I, II, III, IV e V são verdadeiras.
- d) II, III e IV são verdadeiras.
- e) IV é verdadeira.

Normas e modelos de qualidade

- A utilização de técnicas de gestão da qualidade, com novas tecnologias de *software* e métodos de teste, conduziu a melhorias significativas no nível de qualidade do *software* nos últimos vinte anos (Sommerville, 2016).
- “A qualidade é importante, mas, se o usuário não está satisfeito, nada mais importa.”



**Satisfação do usuário = produto adequado
+ máxima qualidade
+ entrega dentro do orçamento e do cronograma**

Principais objetivos da qualidade:

- Reunir um conjunto de características, regras e procedimentos de modo que o *software* satisfaça as necessidades de seus usuários;
- Levar menos tempo para fazer uma coisa correta do que explicar o porquê foi feito errado;
- Garantir a qualidade mesmo depois de estar pronto o produto;
- Estabelecer cronogramas e custos reais.

SPICE – ISO/IEC 15504: características

- A ISO/IEC 15504, conhecida como SPICE – *Software Process Improvement & Capability dEtermination* (Melhoria do Processo de *Software* e Determinação da Capacidade), é uma norma internacional para avaliar a capacidade e a maturidade dos processos de desenvolvimento de *software* em uma organização.
- A SPICE – ISO/IEC 15504 tem base nos modelos já existentes como ISO 9000 e CMMI. Ela é uma “evolução” da ISO/IEC 12207, que possui níveis de capacidade para cada processo.

O framework da ISO 15504 inclui um modelo de referência, que serve de base para o processo de avaliação. Um conjunto de processos estabelecidos em duas dimensões:

- Dimensão de processo.
- Dimensão de capacidade.

SPICE – ISO/IEC 15504: *framework*

Na dimensão de processo, o modelo é dividido em cinco grandes categorias:

- Cliente-Fornecedor.
- Engenharia.
- Suporte.
- Gerência.
- Organização.

Processos Primários

Aquisição (*Acquisition*)

- ACQ. 1: preparação
- ACQ. 2: seleção de fornecedor
- ACQ. 3: contrato
- ACQ. 4: monitoração do fornecedor
- ACQ. 5: aceitação

Fornecimento (*Supply*)

- SPL. 1: proposta (*tendering*)
- SPL. 2: *release* do produto
- SPL. 3: aceitação e suporte

Engenharia (*Engineering*)

- | | |
|--------------------------|-----------------------------------|
| ENG. 1: elic. requisitos | ENG. 7: integração de SW |
| ENG. 2: an. req. sistema | ENG. 8: teste de SW |
| ENG. 3: arq. sist. | ENG. 9: integração de sistema |
| ENG. 4: an. req. SW | ENG. 10: teste de sistema |
| ENG. 5: design SW | ENG. 11: instalação de SW |
| ENG. 6: constr. SW | ENG. 12: manutenção de SW e sist. |

Operação (*Operation*)

- OPE. 1: uso operacional
- OPE. 2: suporte à operação

Processos Organizacionais

Gestão (*Management*)

- MAN. 1: alinhamento organizacional
- MAN. 2: gestão organizacional
- MAN. 3: gestão de projeto
- MAN. 4: gestão da qualidade
- MAN. 5: gestão de risco
- MAN. 6: medição

Melhoria de Processos (*Process Improvement*)

- PIM. 1: definição de processo
- PIM. 2: avaliação de processo
- PIM. 3: melhoria de processo

Gestão de Recursos (*Resource and Infrastructure*)

- RIN. 1: recursos humanos
- RIN. 2: treinamento
- RIN. 3: gestão do conhecimento
- RIN. 4: infraestrutura

Reúso (*Reuse*)

- REU. 1: gestão de ativos (*assets*)
- REU. 2: gestão do programa de reúso
- REU. 3: engenharia de domínio

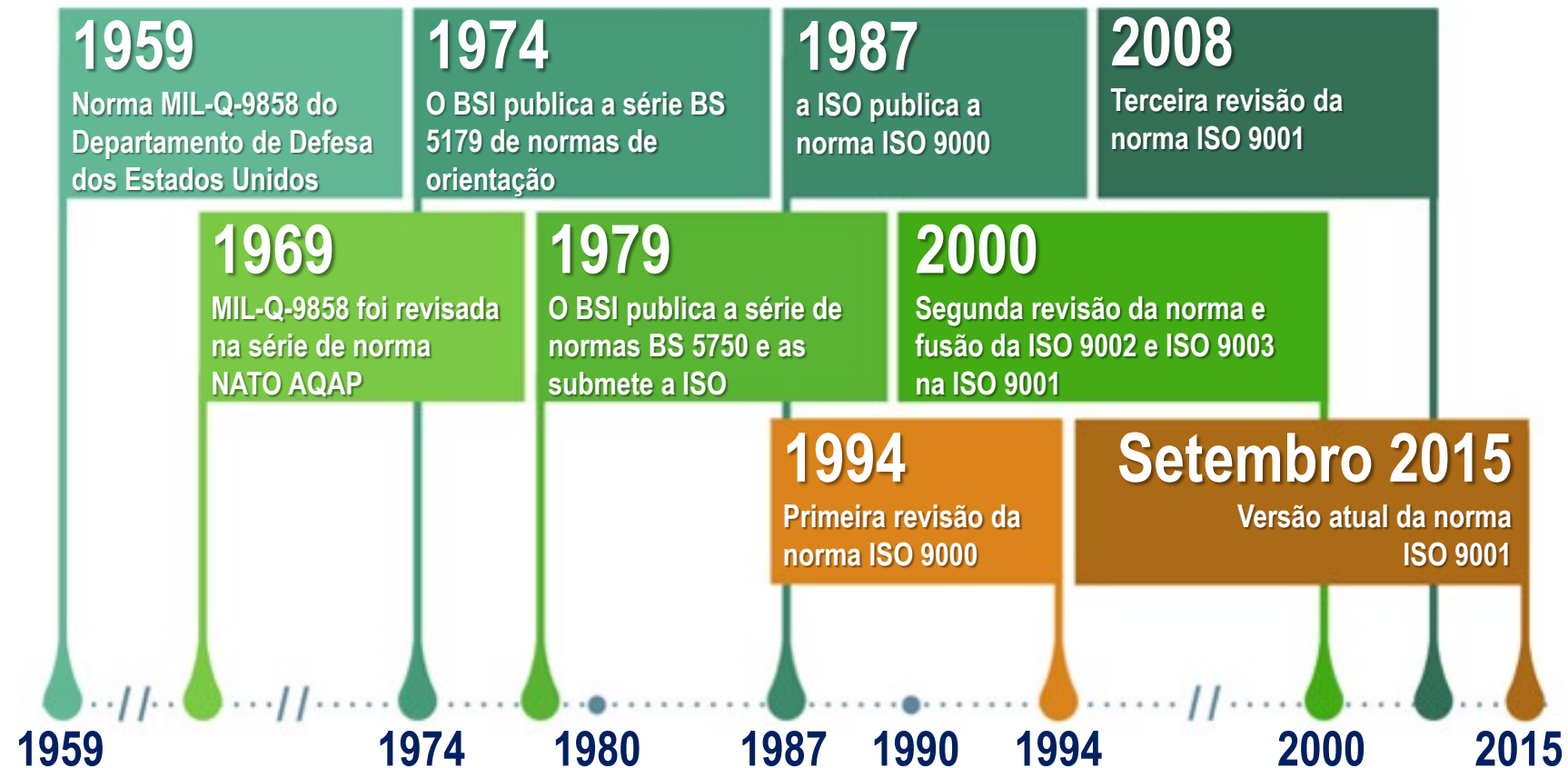
Processos de Apoio

Apoio (*Support*)

- | | |
|-------------------------------|--------------------------------|
| SUP. 1: garantia da qualidade | SUP. 6: avaliação de produto |
| SUP. 2: verificação | SUP. 7: documentação |
| SUP. 3: validação | SUP. 8: gestão de configuração |
| SUP. 4: revisão conjunta | SUP. 9: solução de problemas |
| SUP. 5: auditoria | SUP. 10: gestão de mudança |

ISO 9000: características do sistema da qualidade

- A ISO 9000 é uma série de normas internacionais que estabelecem diretrizes e requisitos para sistemas de gestão da qualidade em organizações de diversos setores.
- Essas normas foram desenvolvidas pela *International Organization for Standardization* (ISO) e são amplamente reconhecidas e adotadas em todo o mundo.
- No modelo ao lado, um breve histórico das normas de Sistemas de Gestão da Qualidade.

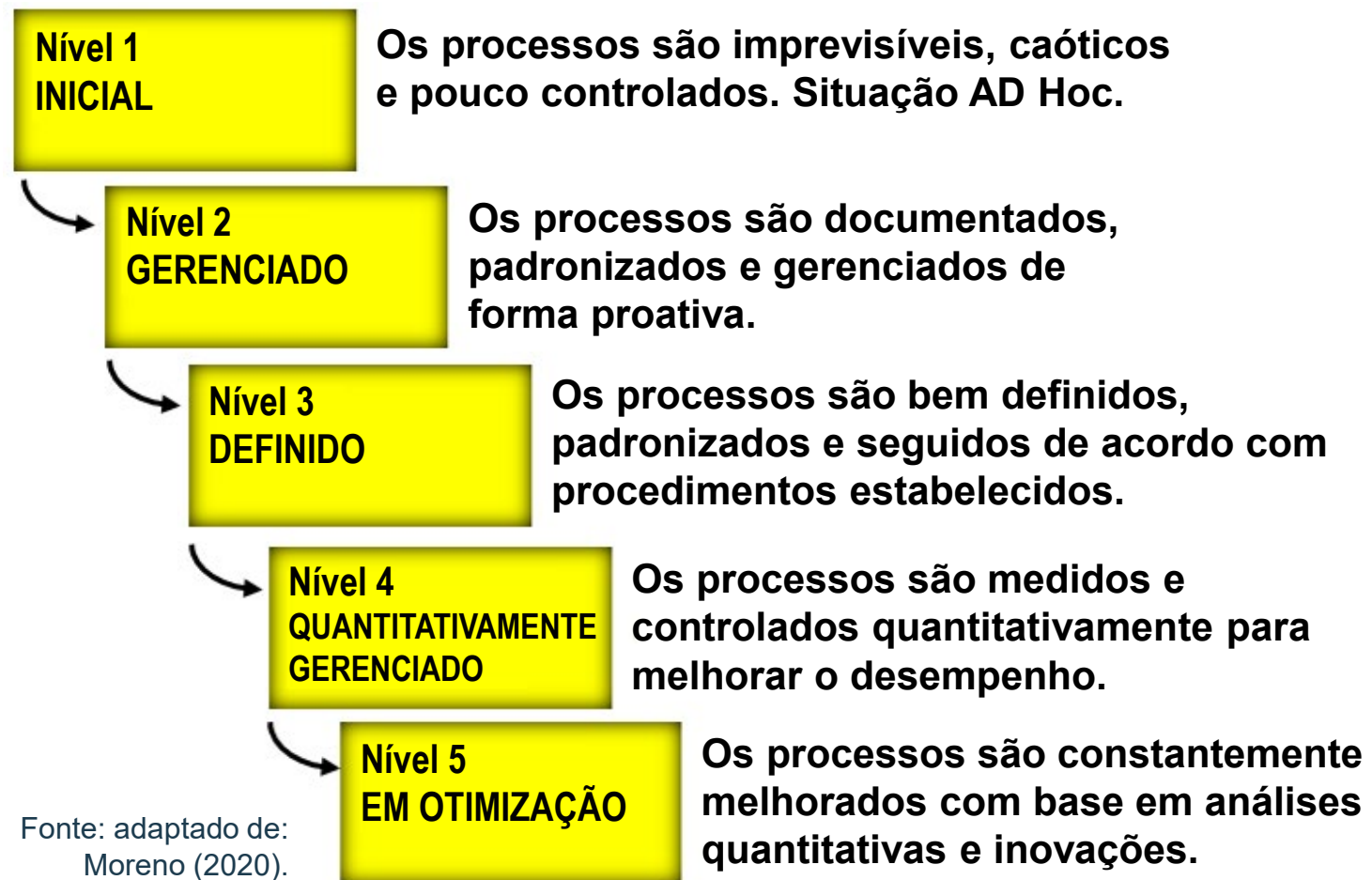


ISO 9000 e as relações com a ISO 9001

- A ISO 9000 é definida como “Padrões para a Gerência da Qualidade e Garantia de Qualidade. Diretrizes para a Seleção e Uso”.
- A ISO 9000 orienta as organizações a desenhar seus sistemas de qualidade.
- A ISO 9000 é uma norma composta por várias normas, sendo a ISO 9001 (modelo de garantia de qualidade em projeto, instalação, desenvolvimento, produção, arquitetura e serviço) a mais conhecida e utilizada.
 - A ISO 9001 define os requisitos para implementação de um sistema de gestão da qualidade (SGQ) que aumente a satisfação dos clientes em conformidade com as expectativas e requisitos do cliente.
 - Na engenharia de *software*, empresas que possuem certificação ISO 9001 demonstram capacidade para atender os requisitos dos clientes.

CMMI: modelo integrado de maturidade e capacidade

- O CMMI – *Capability Maturity Model – Integration* (Modelo de Maturidade em Capacitação – Integração) é um modelo de melhoria de processos desenvolvido para auxiliar organizações a aprimorarem sua capacidade de desenvolvimento e gerenciamento de sistema de *software*.
- O modelo integrado de maturidade e capacidade descreve uma estrutura de capacidade em 5 níveis como mostra a figura.



CMMI: KPA – Key Process Area

- O CMMI determina práticas recomendadas em Áreas-Chave de Processo – ACP (do inglês, *Key Process Area – KPA*), como mostra o quadro ao lado.

Fonte: adaptado de: Moreno (2017).

Gerencial	Organizacional	Eng. de software
Nível 1 – Inicial (Ad Hoc)		
Nível 2 – Repetitivo		
Controle de Projeto Gerência: Configuração Gerência: Requisitos Gerência: Subcontratação Medição e Análise Planejamento do Projeto Garantia de Qualidade		
Nível 3 – Definido		
Análise de Decisões Gerência: SW – Integração	Definição do processo Foco no processo Programa de Treinamento Validação	Desenvolvimento: Requisitos Gerência: Risco Integração do produto Solução dos requisitos Verificação
Nível 4 – Gerenciado		
Gerência: Quantitativa	Desempenho do processo	
Nível 5 – Otimização		
	Gestão do processo	Análise e Resolução

Interatividade

O CMMI (*Capability Maturity Model Integration*) é o modelo de maturidade da qualidade do *software* recomendado no desenvolvimento. Esse modelo tem por objetivo unificar e agrupar diferentes normas e padrões de modelos anteriores. O modelo CMMI determina práticas recomendadas chamadas de:

- a) CASE – *Computer-Aided software Engineering* (Engenharia de *Software* Apoiada por Computador).
- b) KPA – *Key Process Area* (Áreas-Chave de Processo).
- c) RUP – *Rational Unified Process* (Processo Unificado da Rational).
- d) SPICE – *Software Process Improvement & Capability dEtermination* (Melhoria do Processo de *Software* e Determinação da Capacidade).
- e) UML – *Unified Modeling Language* (Linguagem Unificada de Modelagem).

Resposta

O CMMI (*Capability Maturity Model Integration*) é o modelo de maturidade da qualidade do *software* recomendado no desenvolvimento. Esse modelo tem por objetivo unificar e agrupar diferentes normas e padrões de modelos anteriores. O modelo CMMI determina práticas recomendadas chamadas de:

- a) CASE – *Computer-Aided software Engineering* (Engenharia de Software Apoiada por Computador).
- b) KPA – *Key Process Area* (Áreas-Chave de Processo).
- c) RUP – *Rational Unified Process* (Processo Unificado da Rational).
- d) SPICE – *Software Process Improvement & Capability dEtermination* (Melhoria do Processo de Software e Determinação da Capacidade).
- e) UML – *Unified Modeling Language* (Linguagem Unificada de Modelagem).

Referências

- BECK, Kent. *et al. Manifesto para Desenvolvimento Ágil de Software*. 2001. Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 30 jun. 2020.
- CHEESMAN, J.; DANIELS, J. *UML components: a simple process for specifying component based software*. Addison-Wesley, 2000.
- CÔRTEZ, Mario L. *Modelos de qualidade de software: ISO 15504*. São Paulo: IC-UNICAMP, 2017.
- 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.

Referências

- PMBOK®. *Um guia do conhecimento em gerenciamento de projetos* – Guia PMBOK®. 6. ed. Atlanta: Project Management Institute (PMI), 2017.
- PRESSMAN, R. S. *Engenharia de software*. 7. ed. São Paulo: McGraw-Hill, 2011.
- SLACK, Nigel. *Administração da produção*. São Paulo: Atlas, 2006.
- SOMMERVILLE, Ian. *Engenharia de software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011.
- SOMMERVILLE, Ian. *Software engineering*. 10. ed. Pearson Education Limited, 2016.
 - STOJANOVIC, Strahinja. *Versão ISO 9001:2015* – Lista de materiais úteis. Advisera. Disponível em: <https://advisera.com/9001academy/pt-br/knowledgebase/versao-iso-90012015/>. Acesso em: 25 jul. 2023.

ATÉ A PRÓXIMA!