# Unidade II

## 5 COMPONENTIZAÇÃO E REÚSO DO 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, 2002).

O componente de software tem características únicas, possíveis de serem implantadas e substituíveis em um sistema, que encapsula a implementação e exibe o conjunto de interfaces. O componente de software representa um conjunto de programas (ou classes), uma estrutura endereçável e independente que representa uma função específica.

Como pode ser visto na figura 29, na modelagem de acordo com o padrão UML existem duas representações simbólicas do componente de software usados: o bloco da esquerda tem formato simples, a representação do bloco é feita com uma caixa, dois tabs e o nome do componente, que pode ser usado em projetos e uma arquitetura de sistema/software; e a direita tem formato com especificação de atributos, usado em projetos para programação.

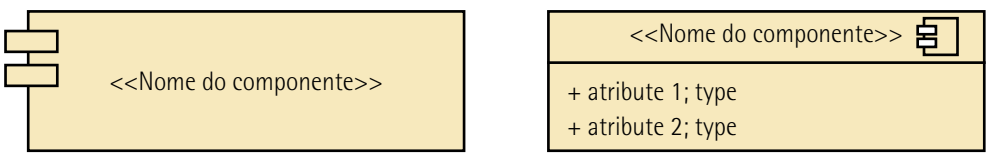


Figura 29 – Componentes de software

O módulo é um atributo individual do software que permite gerenciar apenas um programa, um software ou um sistema. A modularidade do software visa uma interpretação simples do projeto, que permite boas análises para o suporte e a manutenção do sistema.

O módulo pode ser representado de diversas formas, desde desenhos simples até formas mais complexas por meio de diagramas, que visam o projeto do sistema/software.

Na modelagem, a figura 31 apresenta um bloco construtivo que, de acordo com o padrão UML, representa um módulo de implantação (do inglês, *deployment*) e que pode ser chamado também de módulo de distribuição.

O bloco de implantação, é representado por uma caixa com faixas de sombreados no topo e à direita, dando a noção de 3D. Esse símbolo representa um "nó", identificado pelo nome do bloco de implantação.

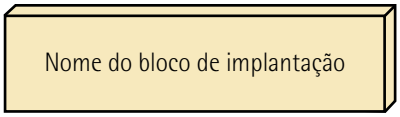


Figura 30 – Bloco de Implantação

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 ou software, ou parte dele, como um componente. Pode ser um computador, um dispositivo móvel, uma estrutura de memória ou mesmo um dispositivo periférico.

**Observação**

Modularidade de um sistema ERP

O ambiente operacional padrão do ERP se baseia em uma arquitetura servidor/cliente. Em um desenho simples, a figura 31 representa a arquitetura servidor/cliente como um módulo de implantação dos requisitos do sistema (RS), necessários para o ERP.

Com o desempenho necessário permite-se a implantação de vários componentes de software no módulo ERP. Essa técnica favorece os negócios porque os módulos ou componentes podem ser implantados de acordo com as necessidades do cliente. Observe a figura 31, os produtos de software disponibilizados em módulos, possíveis de serem configurados no mesmo ambiente operacional.


Figura 31 – Módulos e componentes de um sistema ERP da empresa softsystem.com

Disponível em: http://tinyurl.com/5n6n88vs. Acesso em: 2 jan. 2023.

A principal diferença entre um módulo e um componente de software é a sua reutilização e independência. O módulo é a parte interna de um sistema de software que ajuda na organização do código, mas não é destinado à reutilização independente.

Observe na figura 31 o módulo corresponde ao ERP, que funciona basicamente em uma arquitetura servidor/cliente com recursos de coesão e acoplamento (vistos no tópico 1.3.1 Modularidade: coesão e acoplamento) para integrar os diversos componentes de software.

O componente de software é uma unidade autônoma e reutilizável que pode ser integrada em diferentes sistemas ou aplicativos, promovendo a modularidade e a reutilização de código. Na figura 31, o componente de software corresponde aos demais serviços integrados ao ERP, tais como: relatório em tempo real, aumento dos lucros, melhoria nos processos, gestão de compras, redução de custos, gestão de vendas e demais outros apresentados.

São frequentemente usados em desenvolvimento de software orientado a componentes e em Arquiteturas Orientada a Serviços (do inglês *Service-Oriented Architecture* – SOA). SOA é uma abordagem em que serviços menores (microsserviços) independentes e especializados são criados e implementados no sistema de software.

Os componentes de software devem ser produzidos de modo que possam ser reusados, ou seja, que possam ser adaptados aos módulos, a novas plataformas e ambientes operacionais. O reúso de um componente é uma atividade natural no processo de engenharia.

## 5.1 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 ter uso em outras aplicações. Ela se manifesta à medida que são produzidos componentes de software, que podem ser combinados de forma lógica para produzir outros novos componentes, módulos ou mesmo novos produtos de software.

A prática da diversificação é adquirir e construir um repertório de alternativas, isto é, a matéria-prima do projeto: componentes, soluções de componentes e conhecimento. A figura 32 fornece diversos componentes e módulos disponíveis para o projetista construir e implantar infraestruturas de tecnologia da informação que deem apoio a um determinado sistema.

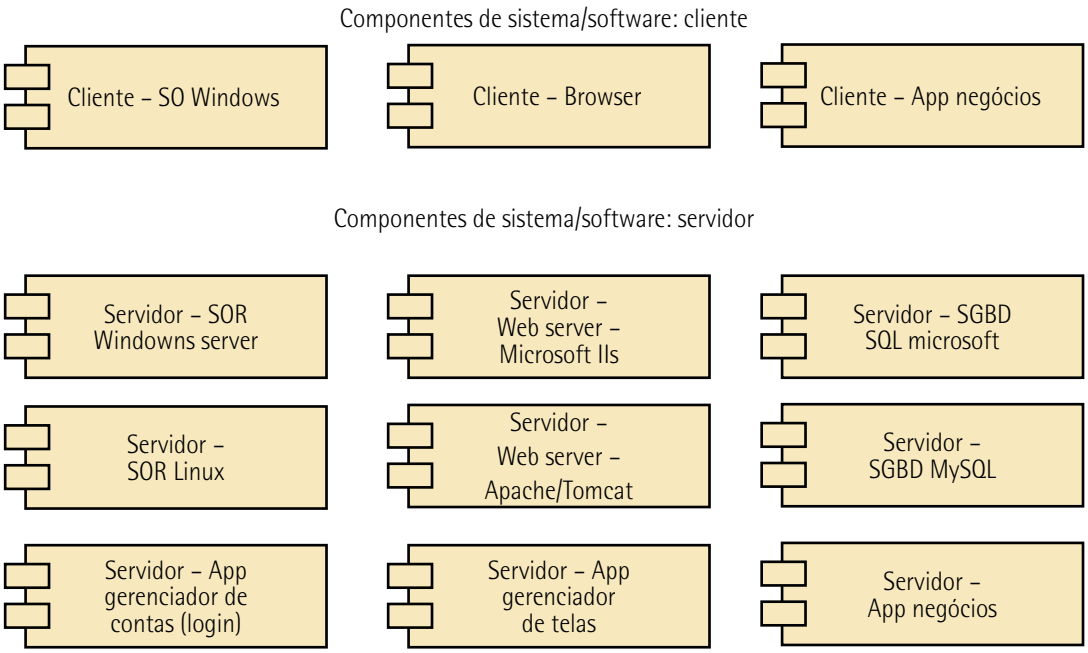


Figura 32 – Repertório de alternativas de componentes e módulos para a construção de uma infraestrutura de TI para diversos sistemas

A engenharia de domínio objetiva identificar, construir, catalogar e disseminar um conjunto de componentes de software que tenham aplicabilidade para o software já existente e os que venham a ser criados, dentro de um domínio de aplicação específico. Quando bem elaborada, permite que as peças (componentes) produzidas fiquem à disposição para serem usadas em outros projetos.

Um domínio de aplicação é como uma família de produtos, as aplicações têm funcionalidade (ou intenção de funcionalidade) similar. O objetivo é estabelecer um mecanismo em que engenheiros de software possam partilhar estes componentes, usando-os em sistemas futuros.

**Observação**

Repositório de componentes para dar suporte à implantação de um sistema BI (*Business Intelligence*):

Além de toda a infraestrutura de TI (repertório mostrado na figura 32), para dar suporte à implantação de um sistema BI são disponibilizados os componentes de software, mostrados na figura 33. Aqui a diversificação também é aplicada. Os componentes apresentados podem ser customizados e ligados para construir o sistema BI. Por exemplo: o BI é o componente principal que necessita do componente Data Mining para seu bom funcionamento. Os demais podem ser empregados para atender os requisitos de interface do software.

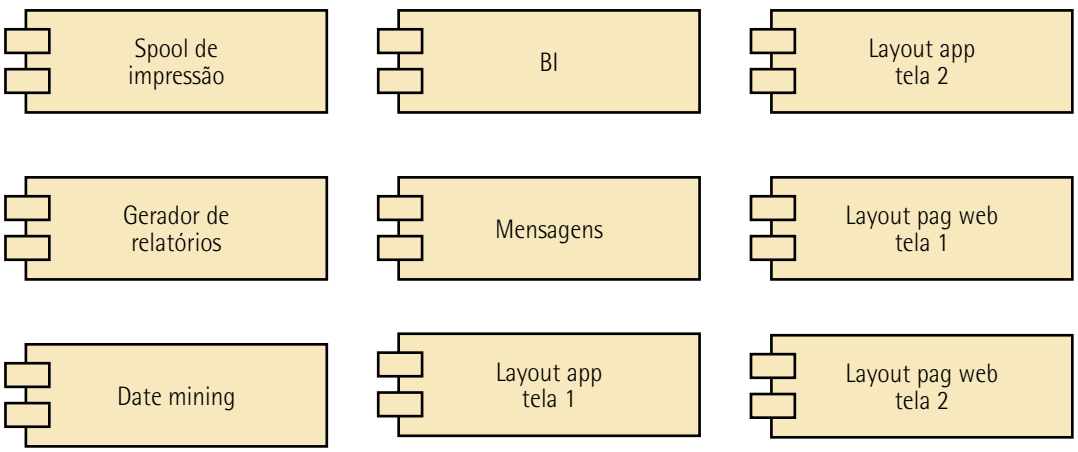


Figura 33 – Repositório de componentes ou blocos funcionais para o suporte da aplicação de um sistema BI

## 5.2 Diagrama de componentes/implantação

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 satisfaçam os requisitos e os resume em uma particular configuração de componentes encapsulada para a criação do produto final. A figura 34 apresenta a convergência de alguns desses componentes em módulos de implantação e a figura 35 mostra apenas a associação de módulos de implantação para o projeto de um sistema.

O diagrama de componentes está associado a todos os requisitos do sistema (RS) pertinentes do projeto, como mostra a figura 34, e pode estar amplamente associado à linguagem de programação que será utilizada para desenvolver o software, à topologia e protocolos de rede, ao SGBD, bem como

interfaces de entradas e saídas. Enfim, todos os recursos de 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. O diagrama de componentes determina como tais componentes estarão estruturados e vão interagir para que o sistema funcione de maneira adequada.

**Saiba mais**

Explore mais o diagrama de componentes. Esta é a principal ferramenta para modelagem de projetos de sistemas de software. Consulte:

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

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 que dão apoio ao software, conforme a figura 34. O diagrama de implantação pode também ser usado para integrar outros subsistemas, associando diversos módulos, como mostra a figura 35.

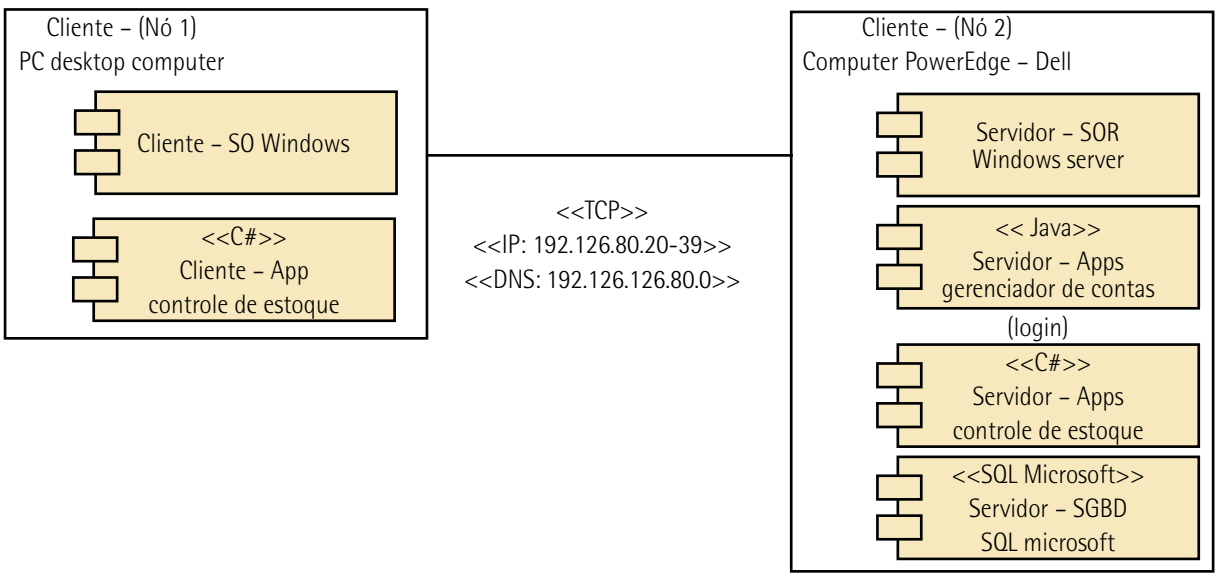


Figura 34 – Diagrama de implantação/componentes de um sistema de controle de estoque

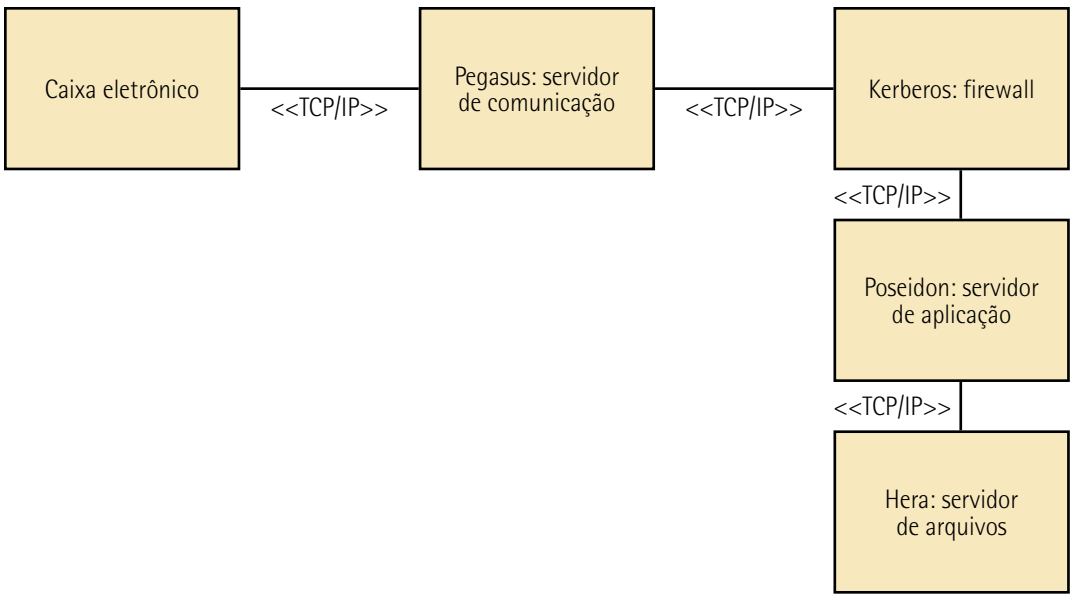


Figura 35 – Estrutura de acesso a caixa eletrônico

Em resumo, o diagrama de implantação encapsula a infraestrutura de TI e/ou a aplicação em que se determinam não só o software, como também as características físicas, como servidores, estações, topologias e protocolos de comunicação, ou seja, todo o aparato físico e funcional sobre o qual o sistema deverá operar. Os diagramas de componentes e de implantação podem ser associados, representados separadamente ou em conjunto.

# 6 GESTÃO E DESENVOLVIMENTO DO SOFTWARE

## 6.1 Fundamentos da gestão de projetos e o guia PMBOK

Um modelo de projeto do software/sistema deverá fornecer detalhes sobre as estruturas de dados, arquitetura, interfaces e componentes do software necessários para implementar o sistema.

Na atualidade, o principal modelo de gestão de projetos amplamente reconhecido e que, apesar de ter suas origens no desenvolvimento de sistemas, é usado pela maioria das empresas em todos os ramos de negócio 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, uma norma reconhecida para a profissão de gerenciamento de projeto. Ele é diferente de uma metodologia. Enquanto uma metodologia é um sistema de práticas, técnicas, procedimentos e regras usadas por aqueles que trabalham numa disciplina, o Guia PMBOK é uma base sobre a qual as organizações podem criar metodologias, políticas, procedimentos, regras, ferramentas e técnicas e fases do ciclo de vida necessários para a prática do gerenciamento de projetos (PMBOK, 2017).

O gerenciamento de projetos:

- Proporciona uma abordagem estruturada e sistemática para planejar, executar e controlar projetos de forma eficiente.
- Ajuda a definir objetivos claros, prazos e recursos necessários, evitando desperdícios e atrasos.
- Permite a identificação e gestão proativa de riscos, reduzindo potenciais impactos negativos.
- Promove a comunicação efetiva entre as equipes, partes interessadas e clientes, garantindo que todos estejam alinhados em relação às expectativas.
- Aumenta a probabilidade de sucesso e satisfação do cliente ao manter o projeto dentro do escopo e do orçamento, além de fornecer uma base para a aprendizagem e melhoria contínua.

De acordo com PMBOK (2017), os projetos têm vários componentes-chave, como são mostrados no quadro a seguir, que se inter-relacionam durante o gerenciamento de um projeto.

**Quadro 12 – Componentes-chave do Guia PMBOK**

| 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 |

Fonte: PMBOK (2017).

O ciclo de vida do projeto pelo PMBOK percorre as fases início do projeto; organização e preparação; execução do trabalho; e terminar o projeto. Detalhes sobre o ciclo de vida do software foram abordados no tópico 2.1 NBR ISO/IEC 12207 – Processos do ciclo de vida do software. Vale a pena dar uma breve lida nesse tópico, pois ele permite ter uma visão analítica de como realizar o projeto do software, como estruturar a organização e suas formas de relacionamento para o desenvolvimento de sistema/software.

No projeto são elaborados vários processos que acompanham e monitoram todas as fases do projeto. Em cada um destes processos existem várias alternativas de componentes, de métodos de montagem e de ferramentas a serem usadas. Todo este aparato deve ser detalhado e estudado minuciosamente com o objetivo de sintetizar as atividades que serão desenvolvidas.

O projeto envolve a redução progressiva do número de alternativas até que o projeto final seja obtido. Observe a figura 36, que expressa bem o que é a ideia do projeto. Na análise do contexto são

filtrados os aspectos de pouca ou nenhuma relevância com o projeto. A abstração dos elementos do projeto permite especificar e modelar os resultados pertinentes ao projeto.

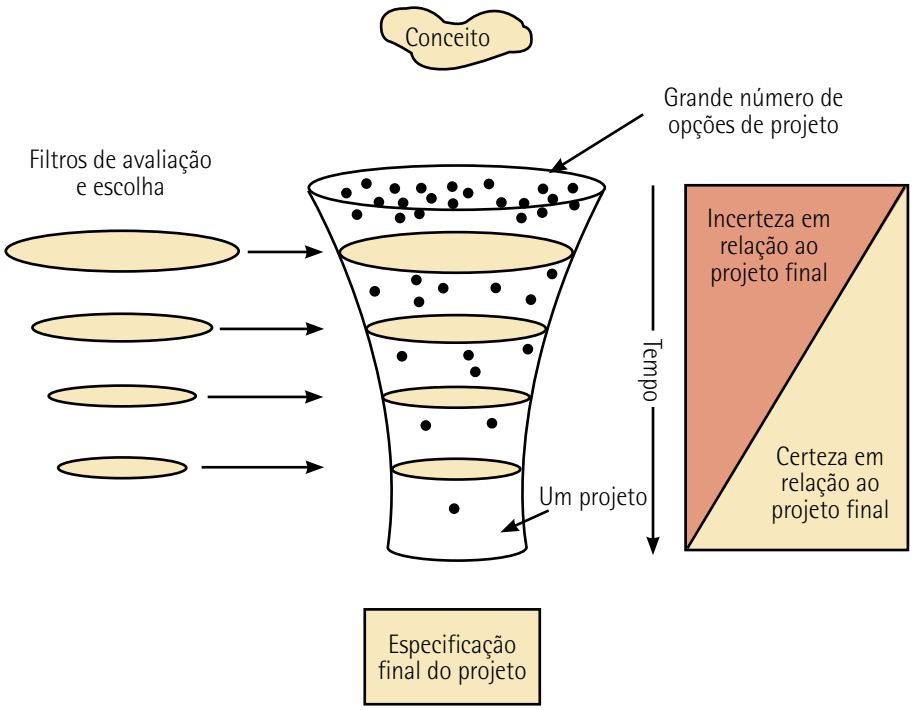


Figura 36 – Características do projeto

Fonte: Slack (2006).

Existem quatro aspectos que devem ser observados ao elaborar o projeto:

1. Criatividade: a imaginação é o aspecto principal de um projeto, pois será criado algo que não existe.

2. Complexidade: o projetista ou a equipe de projeto deverá decidir sobre um grande número de parâmetros e variáveis, que algumas vezes aparecerão conflitantes, que não se encaixam ou que estão fora do orçamento.

3. Compromisso: um cronograma detalhado deve ser elaborado para o acompanhamento do projeto, com o balanço de múltiplos requisitos.

4. Escolha: o projeto exige capacidade de escolha entre diversas soluções e opções para um determinado problema e em todos os níveis, do menor ao maior detalhe, tais como: cores, peças, máquinas a serem usadas, deslocamento de material, resistência do material e segurança.

O ciclo de vida do projeto é gerenciado através da execução de uma série de atividades de gerenciamento de projeto conhecidas como processo de gerenciamento de projetos. Conforme aponta Pressman (2002), um processo é uma sequência de fatos, atividades ou operações que apresentam certa unidade e/ou que se reproduzem com certa regularidade.

Seguindo os princípios do Guia PMBOK (2017), o processo deve ser delimitado definindo seu início e fim. A figura 36 descreve essas delimitações, os pontos de início (input) e fim (output), que em projetos são utilizados como marcos de referências (do inglês *milestones*), úteis para o acompanhamento do processo. Essas marcas servirão para controlar o processo, medir, avaliar, projetar melhorias e tendências.

A figura 37 mostra o modelo básico de um processo, com suas respectivas marcas de referência: input, processo e output.

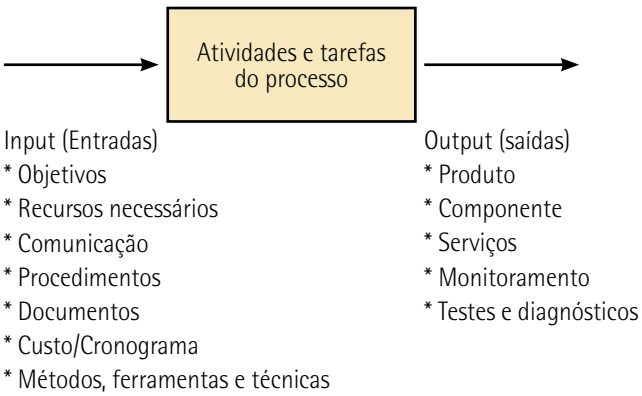


Figura 37 – Modelo básico do processo

- **Input**: o ponto inicial, normalmente marca os itens necessários para dar entrada no processo. De maneira geral, esses itens referem-se aos requisitos de entrada do processo, tais como: objetos claros e específicos, recursos humanos necessários, materiais, tecnologia a ser empregada, fornecedores, componentes do software/sistema, metodologias, ferramentas, custo, cronograma, procedimentos de trabalho e documentos.

- **Processo**: é a combinação organizada dos itens de entrada para criar um produto de software ou de sistema, um componente de outro produto ou executar um determinado trabalho.

- **Output**: é a finalização do processo com a entrega do produto ou serviço determinado no processo. Novos registros são feitos, novas medições e verificações são realizadas, para confrontar o resultado obtido com o objetivo planejado para o processo.

O processo é um conjunto de atividades e tarefas conduzidas por procedimentos de serviços individuais e/ou de equipes, formados com base nos métodos, ferramentas e técnicas combinados.

**Lembrete**

Ao observar o modelo básico do processo, mostrado na figura 36, podemos associar este modelo ao de engenharia de produção, em que input corresponde aos recursos a serem transformados, o processo corresponde aos recursos transformadores e output corresponde ao produto gerado.

Os grupos de processos que acompanham todo o ciclo de vida do PMBOK são:

- **Processos de iniciação**: são realizados para definir um novo projeto ou uma nova fase de um projeto existente. Nesta etapa são feitas análises de viabilidade do projeto e levantamento dos requisitos de escopo dele.

- **Processos de planejamento**: é definido um plano detalhado com os objetivos, escopo, atividades, recursos e cronograma do projeto. Inclui também identificação de riscos, estratégias de mitigação e comunicação com as partes interessadas. O resultado é um plano abrangente que orienta a execução do projeto e ajuda a alcançar os resultados desejados com sucesso.

- **Processos de execução**: é a fase em que o trabalho planejado é realizado para satisfazer os requisitos do projeto. O processo de execução transforma o planejamento em resultados concretos e bem-sucedidos.

- **Processos de monitoramento e controle**: é uma fase crítica do gerenciamento de projetos. É realizado o controle das atividades e o gerenciamento de mudanças, visando manter o projeto dentro do escopo, prazo e orçamento previamente definidos. O processo de monitoramento e controle permite tomar ações corretivas proativas, minimizando riscos e garantindo que o projeto siga na direção certa, em conformidade com os objetivos estabelecidos.

- **Processos de encerramento**: nesta etapa, o projeto é formalmente concluído e as entregas são verificadas e aceitas pelos stakeholders. São finalizadas todas as atividades pendentes, encerrados os contratos e liberados os recursos. O processo de encerramento inclui também a elaboração de relatórios de lições aprendidas e a celebração do sucesso do projeto. A fase de encerramento é fundamental para garantir que o projeto seja finalizado de forma controlada, documentando resultados e aprendizados para projetos futuros.

Além de grupos de processos, os processos também são categorizados por áreas de conhecimento. A área de conhecimento é uma área identificada de gerenciamento de projetos, definida por seus registros de conhecimento e descrita em termos dos processos que a compõem: práticas, entradas, saídas, ferramentas e técnicas (PMBOK, 2017).

Veja a figura 38. São basicamente dez as áreas de conhecimento que identificam as principais gerências atuantes no projeto, de acordo com o PMBOK (2017).

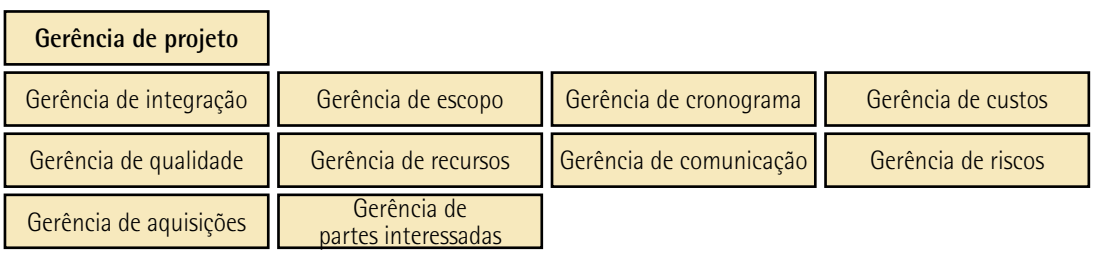


Figura 38 – Áreas de conhecimento e gerenciamento de projetos

Fonte: PMBOK (2017).

A partir daí cada área de conhecimento (ou gerência) é associada a um ou mais processos do grupo de processos, de influência no projeto. Essas relações estão descritas em detalhes no Guia PMBOK.

Exemplo: associação da gerência do Escopo do Projeto (área de conhecimento) com seus respectivos processos, de acordo com o PMBOK:

Área de conhecimento: gerenciamento do escopo do projeto.

Processos associados:

1. Grupo de processos de planejamento: planejar o gerenciamento do escopo; coletar os requisitos; definir o escopo; e criar a EAP (Estrutura Analítica do Projeto).

2. Grupo de processos de monitoramento e controle: validar e controlar o escopo.

## 6.2 Metodologias ágeis

### 6.2.1 Manifesto para Desenvolvimento Ágil de Software

O *Manifesto for Agile Software Development* (Manifesto para Desenvolvimento Ágil de Software) foi publicado nos dias 11 a 13 de fevereiro de 2001, por Kent Beck e mais dezesseis notáveis desenvolvedores, que se reuniram para defender as regras que reproduzimos a seguir:

> **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.
>
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.
>
> Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas (Beck *et al.*, 2001).

Um manifesto é um texto ou declaração pública que expressa princípios, ideias ou objetivos de um indivíduo, grupo ou movimento. Geralmente escrito de forma impactante e persuasiva, o manifesto busca influenciar a opinião pública, incitar mudanças sociais ou políticas e promover adesão a uma causa. É basicamente o que busca o desenvolvimento ágil.

Observe a figura 39. 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.

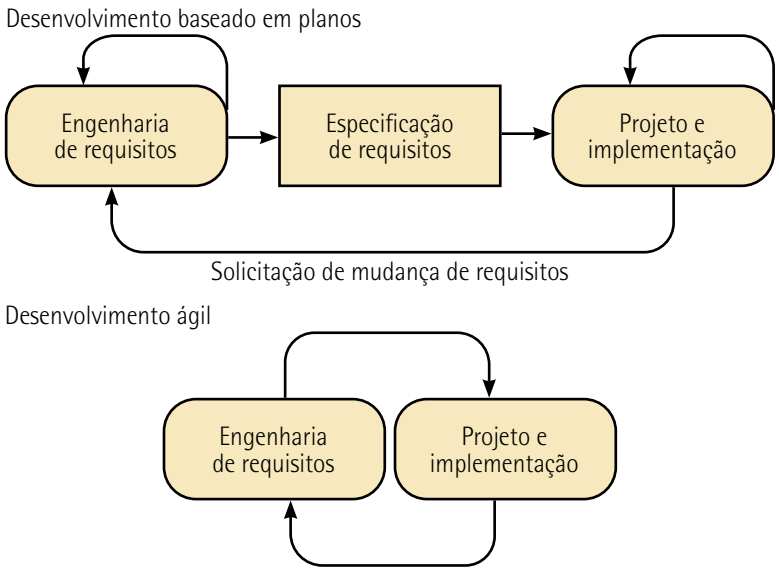


Figura 39 – Especificação ágil dirigida a planos

Fonte: Sommerville (2016).

Já o desenvolvimento ágil é uma abordagem iterativa e incremental, com baixo índice de especificação, que prioriza a adaptação a mudanças e a entrega de valor contínuo ao cliente.

O prescritivo enfatiza a previsibilidade e o planejamento detalhado, enquanto o ágil valoriza a flexibilidade, colaboração e feedback constante. O desenvolvimento prescritivo pode ser mais adequado para projetos estáveis e bem definidos, enquanto o ágil se destaca em cenários dinâmicos e sujeitos a mudanças constantes.

A frase do manifesto "mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda" (Beck, 2001) pode ser explicada com base na figura 40. 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.

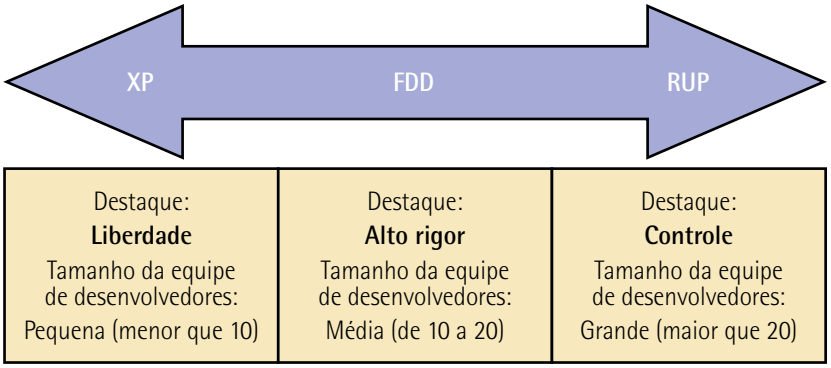


Figura 40 – Valores da metodologia ágil

**Saiba mais**

Veja a publicação do Manifesto para Desenvolvimento Ágil de Software.

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

### 6.2.2 Principais metodologias ágeis

Nesse tópico teremos uma visão geral sobre algumas das aplicações das principais metodologias ágeis:

- **XP – *Extreme Programming* (do inglês programação extrema)**: aplicada em equipes pequenas de programação.
- **Scrum**: equipes de desenvolvimento com foco de construir e integrar software em ambientes complexos, com requisitos pouco estáveis, desconhecidos ou que mudam com frequência.
- **FDD – *Feature Driven Development* (do inglês desenvolvimento dirigido a funcionalidades)**: apropriado para implementação de funcionalidades em projetos de médio e grande porte.
- **DSDM – *Dynamic Systems Development Method* (do inglês 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 (do inglês modelagem ágil)**: prática para modelagem e documentação do software.

Para mostrar como funcionam as metodologias ágeis, destacamos nesse tópico a metodologia ágil XP – *Extreme Programing*. As ideias subjacentes aos métodos ágeis foram desenvolvidas mais ou menos ao mesmo tempo por várias pessoas diferentes na década de 1990. No entanto, talvez a abordagem mais significativa para mudar a cultura de desenvolvimento de software foi o desenvolvimento do *Extreme Programming* (XP) Sommerville (2016).

O nome XP – *Extreme Programing* foi cunhado por Kent Beck em 1998 (*apud* Sommerville, 2016) porque a abordagem foi desenvolvida levando boas práticas reconhecidas, como o desenvolvimento iterativo, a níveis "extremos". Por exemplo, em XP, várias novas versões de um sistema podem ser desenvolvidas, integradas e testadas em um único dia por programadores diferentes.

A comunicação na XP enfatiza a colaboração estreita entre clientes e desenvolvedores, são estabelecidos termos e conceitos comuns em projeto de forma a criar uma nomenclatura única, feedbacks contínuos e evitar uma documentação volumosa como meio de comunicação.

São projetadas apenas as necessidades imediatas, com projetos simples e de fácil implementação de código. As quatro atividades-chaves da XP a serem desenvolvidas são mostradas no ciclo de desenvolvimento do processo XP, na figura 41.

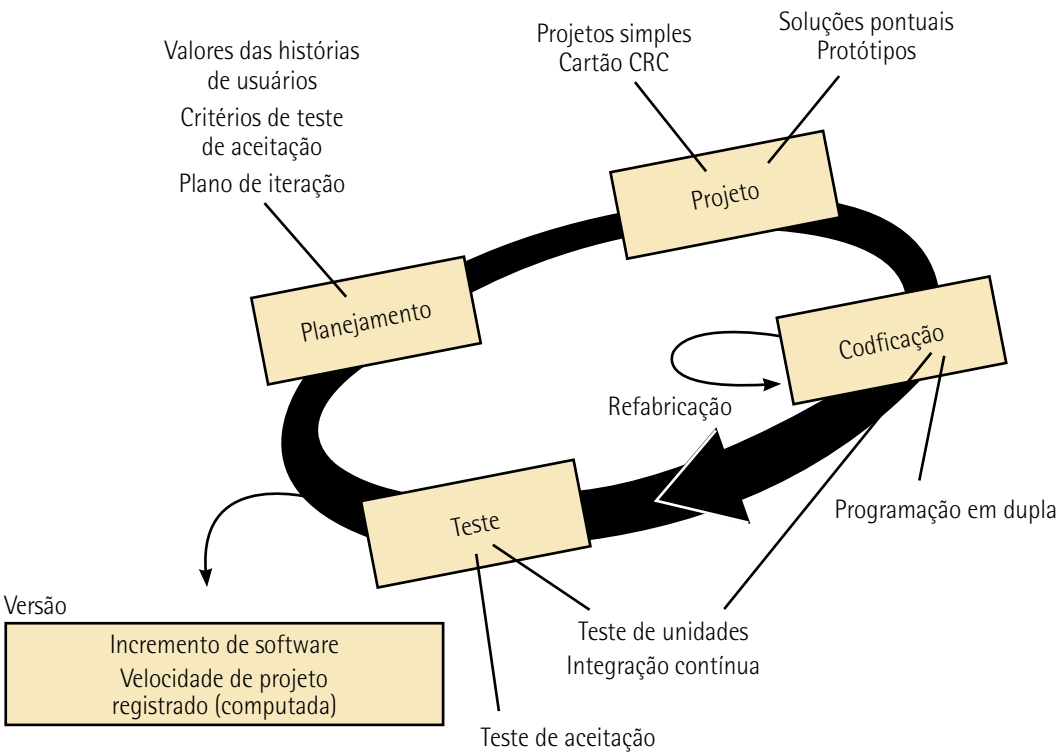


Figura 41 – Atividades-chave da metodologia ágil *Extreme Programming* (XP)

Fonte: Pressman (2011).

Conjunto de regras e práticas da XP:

- **Planejamento**: o planejamento do projeto não é muito elaborado ou algo que leve muito tempo para fazer. Considerando o conhecimento da equipe de desenvolvimento, alguns detalhes são omitidos, tendo em conta a habilidade dos participantes de interpretar os resultados a serem atingidos.

Também chamado de Jogo do Planejamento, nessa fase inicia a atividade "ouvir", em que são levantados os requisitos necessários para que a equipe de desenvolvimento possa entender os resultados solicitados, fatores de qualidade e a funcionalidade a ser desenvolvida.

- **Projeto**: o projeto XP segue o princípio KIS (*keep it simple*, traduzido como preservar a simplicidade). Esse design não comporta muitas funcionalidades com antecedência. Funcionalidades extras, supondo necessidades futuras, são desencorajadas.

Nas ações são usados cartões CRC (classe-responsabilidade-colaborador) para sessões de design. Os cartões CRC identificam e organizam classes orientadas a objetos, suas responsabilidades e colaboradores. Uma atividade de apoio ao uso dos cartões CRC é o Kanban, uma técnica de gestão visual que auxilia na organização e acompanhamento do fluxo de trabalho durante o desenvolvimento do projeto.

Quando problemas são identificados, a XP recomenda a técnica de prototipagem dessa parte do projeto. Assim que os desenvolvedores encontram soluções e melhorias de código, todos os desenvolvedores devem "refatorar" (refazer) o código, que é um método de otimização de projetos. Isso significa que a atividade de projeto é realizada continuamente enquanto o sistema estiver sendo desenvolvido.

- **Codificação**: quando ocorrem as iterações, na implementação do código em vários testes de unidades, todo o código é revisado e os programadores fazem acréscimos de funcionalidades.

A ênfase na XP é a "programação em dupla", em que um dos programadores trabalha na codificação, enquanto o outro cuida da integração, observando os padrões de código e testes das unidades. A ideia é a de que duas cabeças normalmente funcionam melhor que uma.

- **Testes**: a atividade de testes é uma prática contínua e integrada ao processo de desenvolvimento. Testes unitários, de integração e de aceitação são criados para verificar as funcionalidades e detectar falhas o mais cedo possível. Os testes são realizados durante todo o ciclo de vida do projeto para garantir a qualidade do software.

Quando um erro é encontrado, os testes são criados. Registros e casos de testes são revisados para serem repetidos com facilidade. A execução frequente dos testes assegura a estabilidade do sistema e facilita a manutenção contínua. Os testes de aceitação são realizados frequentemente e uma pontuação é executada.

**Lembrete**

Os testes em sua essência correspondem aos processos de apoio verificação e validação (V&V) comentados no tópico 2.1.2 Conceito e classificação dos processos da ISO/IEC 12207.

## 6.3 Normas e modelos de qualidade: SPICE – ISO 15504 E ISO 9000

A gestão da qualidade do software tem por objetivo garantir que os sistemas de software desenvolvidos sejam adequados ao objetivo. Ou seja, os sistemas devem satisfazer as necessidades dos seus utilizadores, ter um desempenho eficiente e confiável e ser entregues a tempo e dentro do orçamento. A utilização de técnicas de gestão da qualidade, juntamente com novas tecnologias de software e métodos de teste, conduziu a melhorias significativas no nível de qualidade do software nos últimos 20 anos (Sommerville, 2017).

### 6.3.1 Modelo de qualidade de software: SPICE – ISO 15504

A ISO/IEC 15504, também conhecida como SPICE – *Software Process Improvement & Capability Determination* (Melhoria do Processo de Software e Determinação da Capacidade) é uma norma internacional desenvolvida para avaliar a capacidade e a maturidade dos processos de desenvolvimento de software em uma organização.

Foi publicada em outubro de 2003, tendo o seu projeto iniciado em 1993 pela ISO com base em modelos já existentes, como ISO 9000 e CMMI. Ela é uma "evolução" da ISO/IEC 12207, que possui níveis de capacidade para cada processo.

A norma ISO/IEC 15504 define um conjunto de práticas e critérios de avaliação divididos em níveis de capacidade, permitindo que as organizações avaliem e classifiquem seus processos de acordo com a eficácia e qualidade. Os níveis de capacidade vão desde o mais baixo (Nível 0 – Incompleto) até o mais alto (Nível 5 – Otimizado), refletindo o grau de excelência e melhoria contínua alcançado na organização.

A adoção da ISO/IEC 15504 ajuda as empresas a identificarem pontos fortes e fracos em seus processos. A norma facilita a comunicação e a colaboração entre as equipes, tornando-se uma ferramenta valiosa para aprimorar os processos de desenvolvimento de software em diversas organizações ao redor do mundo.

O *framework* inclui um modelo de referência, que serve de base para o processo de avaliação. Trata-se de um conjunto de processos fundamentais que orientam para uma boa engenharia de software e estabelece duas dimensões: a de processo e a de capacidade.

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

- cliente-fornecedor;
- engenharia;
- suporte;
- gerência;
- organização.

Na dimensão da capacidade, os níveis de capacitação dos processos são divididos em seis níveis, que vão desde o mais baixo até o mais alto. O objetivo é avaliar a capacitação da organização em cada processo e permitir a sua melhoria. Cada nível representa um estágio de capacidade dos processos da organização em relação a um determinado conjunto de práticas.

O modelo de referência do Spice inclui seis níveis de capacitação dos processos:

- **Nível 0 – Incompleto**: o processo não atinge seus objetivos ou não é executado de forma adequada.
- **Nível 1 – Executado (ou realizado)**: o processo é executado de forma *ad hoc*, sem uma abordagem consistente ou padronizada.
- **Nível 2 – Gerenciado**: o processo é gerenciado de acordo com uma abordagem padronizada e controlada.
- **Nível 3 – Estabelecido**: o processo é executado de maneira definida e padronizada, com ações corretivas quando necessário.
- **Nível 4 – Previsível**: o processo é quantitativamente gerenciado, com metas e medidas de desempenho estabelecidas.
- **Nível 5 – Otimizado**: o processo é continuamente melhorado e otimizado para atender às necessidades em mudança da organização.

A avaliação da capacidade dos processos em cada nível é realizada por meio de evidências objetivas coletadas durante a avaliação, seguindo os critérios estabelecidos pela ISO/IEC 15504.

Os elementos normativos da ISO 15504 – Spice, apresentados na figura 42, estabelecem um plano de análise e desenvolvimento para avaliação do processo.

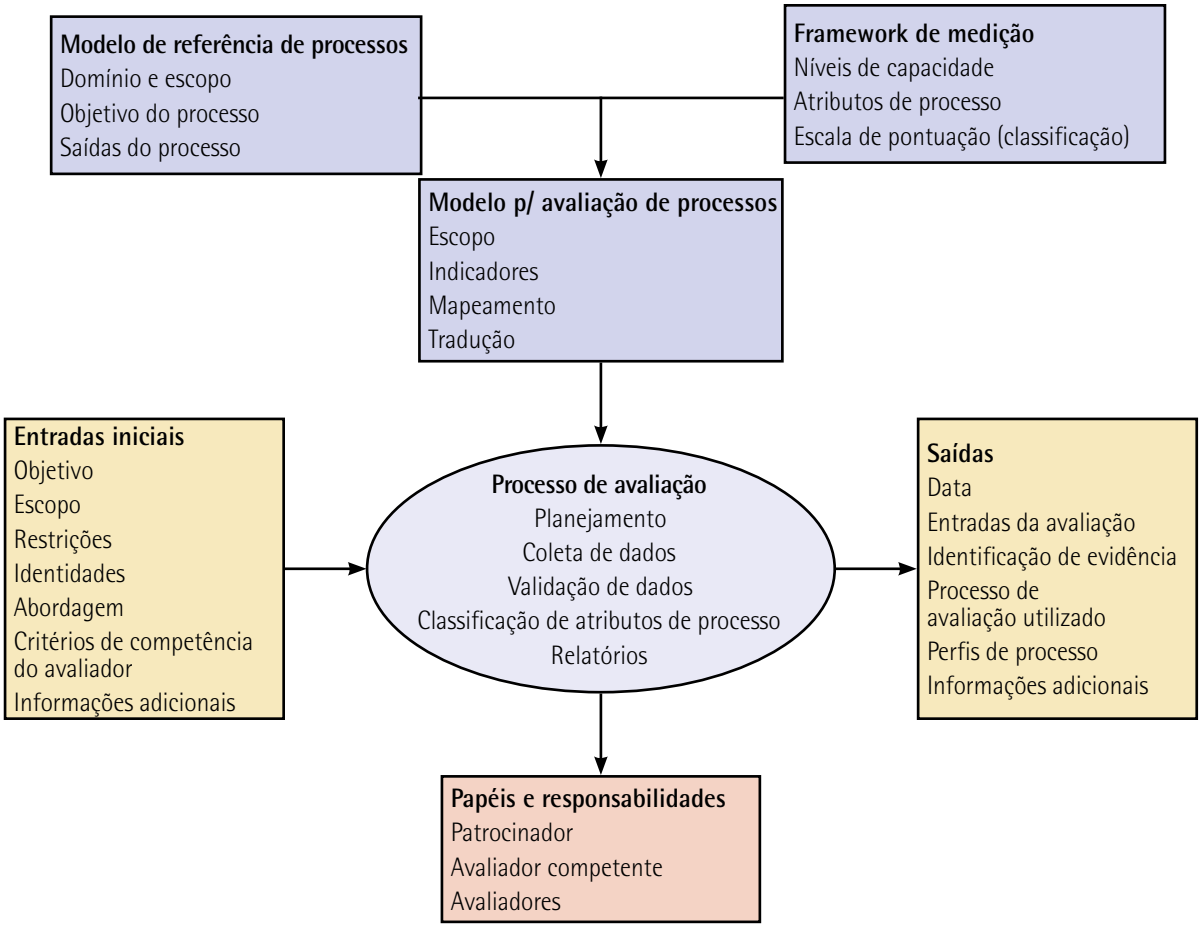


Figura 42 – Elementos normativos da ISO 15504 – Spice

Fonte: Côrtes (2017).

A dimensão dos processos da ISO 15504 – Spice, como mostra a figura 43, apresenta subprocessos e respectivas atividades, nas quais, similar à ISO 12207, também estabelece as áreas de processos: primários, organizacionais e de apoio.

Os subprocessos e atividades variam de acordo com o nível de capacidade, cobrindo desde práticas básicas até práticas otimizadas. Essa dimensão fornece orientações detalhadas sobre como realizar cada atividade para melhorar a eficácia e a eficiência dos processos de desenvolvimento de software na organização, buscando aprimorar a qualidade do software entregue.

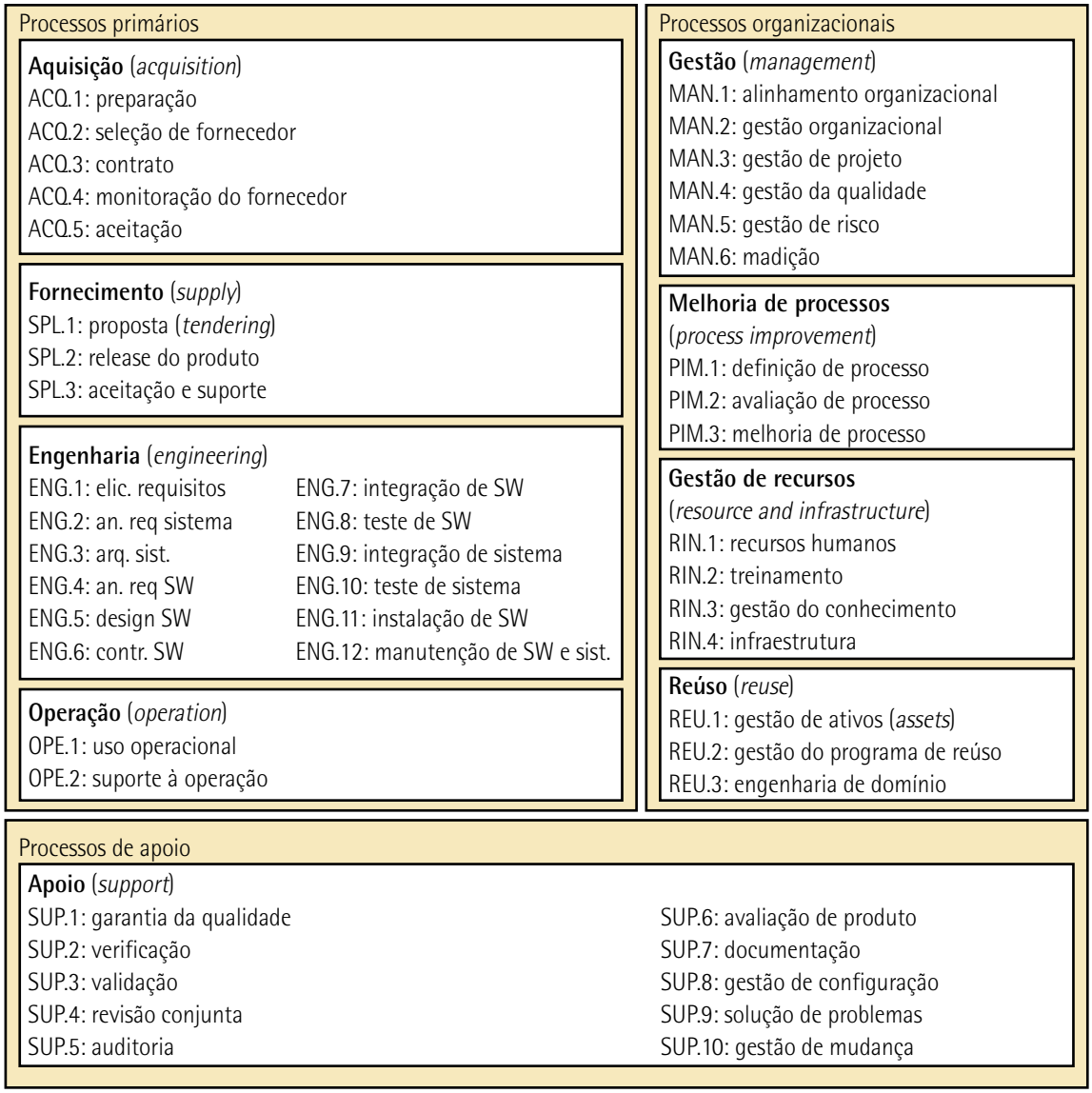


Figura 43 – Dimensão dos processos da ISO 15504 – SPICE

Fonte: Côrtes (2017).

### 6.3.2 Sistemas da qualidade: ISO 9000

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.

De acordo com Stojanovic (2023), a cada 5 a 10 anos a ISO revisa a norma para mantê-la atualizada. A figura 44 mostra um modelo gráfico da história da evolução da ISO 9000.

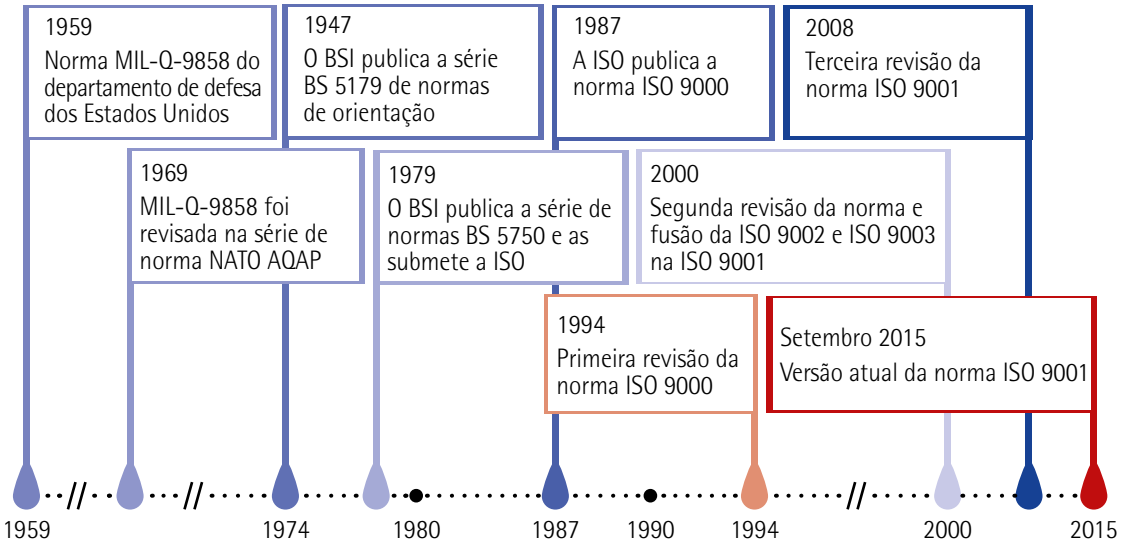


Figura 44 – História das normas de Sistema de Gestão da Qualidade

Fonte: Stojanovic (2023).

A série ISO 9000 é definida como Padrões para a Gerência da Qualidade e Garantia de Qualidade. É uma norma introdutória 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 ligação entre a ISO 9000 e a engenharia de software reside no fato de que a engenharia de software é uma disciplina altamente orientada a processos, que exigem controle rigoroso e garantia de qualidade para o desenvolvimento de sistemas e aplicações de software.

Vemos ainda na figura 44 que no ano 2000 houve a fusão das normas ISO 9002 (Modelo de Qualidade em produção, ensaios e instalação) e ISO 9003 (Modelo de Garantia de Qualidade em inspeção e ensaios finais) na ISO 9001.

A ISO 9001 define os requisitos para implementação de um sistema de gestão da qualidade que visa aprimorar a eficiência dos processos, aumentar a satisfação dos clientes e melhorar a capacidade de entrega de produtos e serviços consistentemente em conformidade com as expectativas e requisitos do cliente.

A ISO 9001 orienta as ações da empresa para a obtenção da Certificação ISO 9001, documento que conduz os esforços da organização para implementar, manter e controlar processos, recursos e pessoas a fim de atender às demandas dos clientes com mais qualidade.

Ao aplicar a ISO 9001 à engenharia de software, uma organização pode estabelecer um sistema de gestão da qualidade para todo o ciclo de vida do desenvolvimento de software, abordando etapas desde a análise de requisitos até a entrega do produto final e suporte contínuo.

Na engenharia de software, empresas que têm certificação ISO 9001 demonstram capacidade de atender os requisitos dos clientes, os regulamentares e os da própria organização, definindo um conjunto de requisitos denominados Sistema de Gestão da Qualidade (SGQ).

## 6.4 Normas e modelos de qualidade: CMMI

*Capability Maturity Model – Integration* – CMMI (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 software, sistemas e produtos.

De acordo com Paulk *et al.* (1993), Watts S. Humphrey (importante pesquisador e pioneiro em engenharia de software) liderou o desenvolvimento do *Capability Maturity Model for Software* – SW-CMM a partir de 1986. Publicado inicialmente pelo Software Engineering Institute (SEI) nos Estados Unidos em setembro de 1987, o SW-CMM foi criado para avaliar a maturidade dos processos de desenvolvimento de software em organizações.

Com base no trabalho de Humphrey e outros pesquisadores, o Capability Maturity Model – CMM (Modelo de Maturidade em Capacitação) foi desenvolvido, abrangendo não apenas o desenvolvimento de software. Paulk e colaboradores definiram a v1.1 do CMM em 1993, abrangendo outras áreas como aquisições e engenharia de sistemas.

A integração entre o CMM e o System Engineering Capability Maturity Model – SE-CMM resultou no Capability Maturity Model – Integration – CMMI. Portanto, o CMMI é a evolução do SW-CMM.

O CMMI permite que as organizações alcancem maior eficiência, qualidade e previsibilidade em seus projetos e processos. Ele fornece uma abordagem estruturada e gradual para avaliar e aprimorar a maturidade dos processos organizacionais. Como mostrado na figura 46, o modelo é baseado em níveis de maturidade, nos quais cada nível representa um grau crescente de otimização e disciplina nos processos.

Os níveis de maturidade do CMMI são os seguintes:

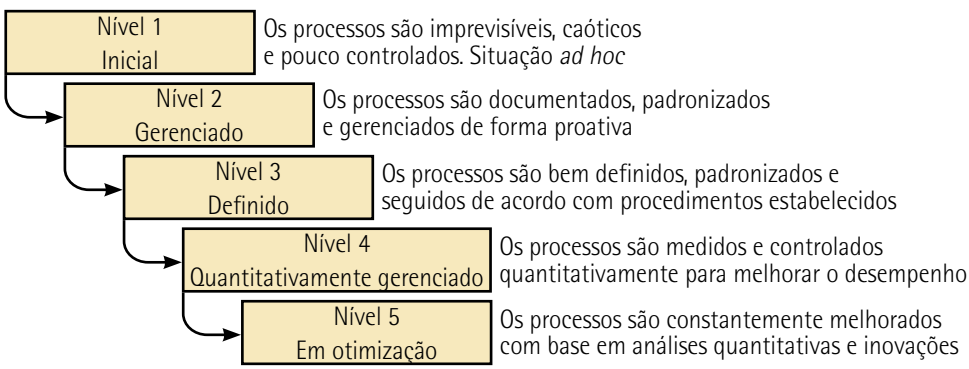


Figura 45 – Os cinco níveis de maturidade do CMMI

Adaptada de: Paulk (1993).

O CMMI determina práticas recomendadas em um certo número de Áreas-chave de Processo – ACP (do inglês *Key Process Area* - KPA), que provê a organização do desenvolvimento de software, baseada nas melhores estratégias para controlar, manter e evoluir os processos administrativos do desenvolvimento de software.

Com base em CMMI (2006), foi montada a tabela abaixo, com as respectivas subgerências (KPAs). A tabela de KPAs é formada por três grandes áreas: Gerencial, Organizacional e Engenharia de Software. Essas áreas gerenciais são alinhadas aos níveis de maturidade do CMMI com suas respectivas KPAs.

**Quadro 13 – KPAs**

| Gerencial<br>Planejamento de projeto de software e gerência | Organizacional<br>Revisão e controle pela gerência sênior | Engenharia de software<br>Especificação, design, codificação e controle de qualidade |
|---|---|---|
| Nível 1 – Inicial (Processo caótico – indica a fase de iniciação da implementação dos processos) | | |
| Nível 2 – Repetitivo (Processo disciplinado) | | |
| Acompanhamento e controle de projeto<br>Gerência de configuração<br>Gerência de requisitos<br>Gerência de subcontratação<br>Medição e análise<br>Planejamento do projeto<br>Garantia de qualidade de processo e produto | | |
| Nível 3 – Definido (processo padronizado e consistente) | | |
| Análise de decisão e resolução<br>Gerência de software integrada | Definição do processo da organização<br>Foco no processo da organização<br>Programa de treinamento<br>Validação | Desenvolvimento de requisitos<br>Gerenciamento de risco<br>Integração do produto<br>Solução técnica dos requisitos<br>Verificação |
| Nível 4 – Gerenciado (processo previsível) | | |
| Gerenciamento quantitativo de processos | Desempenho do processo organizacional | |
| Nível 5 – Otimização (melhoria contínua) | | |
| | Gestão do processo organizacional | Análise e resolução de causas |

## Resumo

A evolução contínua da indústria de software tem levado à busca por abordagens mais eficazes no desenvolvimento de sistemas complexos e de alta qualidade. A unidade II abrange a componentização e o reúso de software como estratégias emergentes e fundamentais para otimizar os processos de desenvolvimento, aumentar a eficiência e reduzir os riscos associados à criação de novos sistemas. Em complemento, no cenário atual de desenvolvimento de software é disposta a gestão eficaz de projetos e a utilização de metodologias adequadas como fundamentais para alcançar resultados bem-sucedidos e produtos de alta qualidade e que atendam às expectativas dos clientes.

Nesse contexto, a unidade II introduz os temas sobre a componentização e reúso do software e a gestão e desenvolvimento do software. Esses temas estão interconectados, pois a criação de componentes reutilizáveis envolve considerações tanto de engenharia de domínio quanto de arquitetura de software. Ao construir componentes com a reusabilidade em mente, as organizações podem acelerar o desenvolvimento de novos sistemas, melhorar a manutenção e reduzir erros recorrentes. Os diagramas de componentes e implantação, por sua vez, fornecem representações visuais cruciais para entender a estrutura e o comportamento de um sistema, auxiliando na comunicação eficaz entre equipes de desenvolvimento e stakeholders.

A abordagem tradicional de desenvolvimento de software, que envolve a criação de sistemas a partir do zero, muitas vezes resulta em altos custos, demoras e dificuldades na manutenção. A componentização se baseia na ideia de dividir um sistema em componentes independentes e bem-definidos, que podem ser reutilizados em diferentes contextos. A engenharia de domínio concentra-se na criação de componentes reutilizáveis em um domínio específico, permitindo a construção eficiente de sistemas por meio da combinação desses componentes.

No cenário atual de desenvolvimento de software, a gestão eficaz de projetos e a utilização de metodologias adequadas são fundamentais para alcançar resultados bem-sucedidos e produtos de alta qualidade. A complexidade crescente das aplicações de software e a demanda por entregas rápidas têm impulsionado a evolução das práticas de gestão e desenvolvimento.

A gestão de projetos de software é um componente crítico para garantir a entrega de produtos que atendam aos requisitos, prazos e orçamentos estabelecidos. O Guia PMBOK é uma referência amplamente reconhecida que reúne boas práticas de gestão de projetos, incluindo aquelas aplicáveis ao desenvolvimento de software. As metodologias ágeis surgiram como resposta à necessidade de se adaptar rapidamente às mudanças frequentes nos requisitos e nas demandas dos clientes. Com a colaboração com o cliente, a adaptação a mudanças, o foco na entrega contínua e a valorização das pessoas, essa abordagem de desenvolvimento de software valoriza a interação direta entre equipes multidisciplinares, ciclos curtos de desenvolvimento (iterações) e a produção de software funcional em intervalos regulares. Entre as principais metodologias ágeis estão Scrum e o *Extreme Programming* (XP). Por fim, a qualidade do software é um objetivo fundamental para garantir a satisfação do cliente e a eficácia das aplicações. Diversas normas e modelos foram desenvolvidos para avaliar e melhorar a qualidade dos processos de desenvolvimento de software. A ISO 15504, também conhecida como Spice, oferece uma estrutura para avaliar a maturidade dos processos. A ISO 9000 é uma norma internacional que estabelece diretrizes para sistemas de gestão da qualidade em várias indústrias, incluindo a de software. E o CMMI fornece um modelo de maturidade que guia as organizações na melhoria contínua de seus processos.

## Exercícios

**Questão 1**. Os componentes do software devem ser produzidos de modo que possam ser reusados, ou seja, que possam ser adaptados a novas plataformas e a novos ambientes operacionais. Considerando esse contexto, avalie as asserções e a relação proposta entre elas.

I – A reusabilidade de software, embora seja uma prática valiosa, tem a desvantagem de sempre aumentar o tempo de desenvolvimento de sistemas.

porque

II – A reusabilidade é uma métrica de qualidade usada para avaliar o quanto um programa, ou parte dele, pode ser usado em aplicações.

Assinale a alternativa correta.

A) As asserções I e II são verdadeiras, e a asserção II justifica a I.

B) As asserções I e II são verdadeiras, e a asserção II não justifica a I.

C) A asserção I é verdadeira, e a II é falsa.

D) A asserção I é falsa, e a II é verdadeira.

E) As asserções I e II são falsas.

Resposta correta: alternativa D.

**Análise das asserções**

I – Asserção falsa.

Justificativa: a reusabilidade de software não necessariamente aumenta o tempo de desenvolvimento de sistemas. Na verdade, quando bem aplicada, a reusabilidade pode levar a economia de tempo e de recursos a longo prazo, já que componentes reutilizáveis podem ser usados em vários projetos.

II – Asserção verdadeira.

Justificativa: a reusabilidade é, de fato, uma métrica de qualidade que avalia o quanto um programa ou seus componentes podem ser reutilizados em diferentes contextos, o que é um indicador de qualidade e de utilidade.

**Questão 2**. (UFU-MG/2023, adaptada) Metodologias de desenvolvimento de software chamadas de ágeis são baseadas em desenvolvimento iterativo, no qual requisitos e soluções evoluem pela colaboração entre equipes auto-organizadas. Essas metodologias encorajam frequente inspeção e adaptação, alinhamento entre o desenvolvimento e os objetivos dos clientes e um conjunto de boas práticas que permita entregas rápidas e de qualidade. Considerando as metodologias ágeis de desenvolvimento de software, avalie as afirmativas.

I – O Scrum adota uma abordagem empírica, entendendo que o problema pode não ser totalmente compreendido ou estar totalmente definido na análise e que os requisitos podem mudar com o passar do tempo. Essa abordagem mantém o foco em maximizar a habilidade da equipe de responder de forma ágil aos desafios emergentes.

II – Um dos valores do método XP (*Extreme Programming*) é a busca pela simplicidade, de modo que o software deve ser mantido o mais simples possível pelo maior tempo possível.

III – O TDD (*Test-Driven Development*) é uma prática que versa sobre pequenas iterações, em que novos casos de teste são escritos considerando a melhoria ou uma nova funcionalidade, e o código necessário é implementado para atender a esse teste.

É correto o que se afirma em:

A) I, apenas.

B) III, apenas.

C) I e II, apenas.

D) II e III, apenas.

E) I, II e III.

Resposta correta: alternativa E.

**Análise das afirmativas**

I – Afirmativa correta.

Justificativa: o Scrum é uma metodologia ágil que enfatiza a adaptabilidade da equipe de desenvolvimento às mudanças. Ele é aplicado a equipes que têm o foco em construir e integrar softwares com requisitos pouco estáveis, desconhecidos ou que mudam com frequência. Ao adotarmos esse método, devemos reconhecer que os requisitos podem evoluir ao longo do tempo e que a compreensão do problema pode ser aprimorada à medida que o projeto avança.

II – Afirmativa correta.

Justificativa: no método XP, são incentivadas a colaboração estreita entre clientes e desenvolvedores e a produção de documentações simplificadas. São projetadas apenas as necessidades imediatas, com projetos simples e de fácil implementação de código.

III – Afirmativa correta.

Justificativa: o TDD é um método de desenvolvimento de software que consiste na conversão de requisitos de software em testes, antes de o software ser concluído. No TDD, casos de teste são escritos antes da implementação do código, e o código é desenvolvido para atender a esses testes. Isso promove o desenvolvimento iterativo e a melhoria contínua do software.