Estratégia PÓS-GRADUAÇÃO | FACULDADE Unyleya

CURSOS DE PÓS-GRADUAÇÃO

MÉTRICAS DE SOFTWARE

Autor:
Ramon Souza

2021

Sumário

1. UNIDADE 1 – Teste de Software

1.1. Introdução ao teste de software

O desenvolvimento de um software pode apresentar falhas. As falhas em um software surgem por diferentes motivos, a exemplo de especificação incompleta ou errônea, requisitos inviáveis de serem implementados pela limitação de hardware ou software, codificação com erro em algoritmo, entre outros motivos.

Nesse contexto, destaca-se o teste de software.

O **teste de software** é um processo ou uma etapa da engenharia de software que possui como principal objetivo revelar falhas no sistema para que sejam corrigidas antes de o produto final ser disponibilizado para o cliente.

O teste de software é uma etapa fundamental no desenvolvimento de um produto de software, pois permite investigar o software a fim de fornecer informações sobre sua qualidade em relação ao contexto que ele deve operar.

1.1.1. Teste no contexto de verificação e validação (V&V)

O teste de software é um elemento de um tópico mais amplo, muitas vezes conhecido como verificação e validação (V&V).

A **verificação** é o conjunto de tarefas que visa garantir que o software implementa corretamente uma função específica. Envolve a análise de um sistema para certificar se este atende aos requisitos funcionais e não funcionais. A questão chave da verificação é: **estamos criando o produto corretamente?**

A **validação** é o conjunto de tarefas que visa assegurar que o software foi criado e pode ser rastreado segundo os requisitos do cliente. Envolve a certificação de que o sistema atende as necessidades e expectativas do cliente. A questão chave da validação é: **estamos criando o produto certo?**

1.1.2. Medidas de um teste

As principais medidas de um teste incluem a cobertura e a qualidade.

A **cobertura** é a medida da abrangência do teste e é expressa pela cobertura dos requisitos e casos de teste ou pela cobertura do código executado**.**

As métricas de cobertura fornecem respostas à pergunta: "O quanto o teste é completo?" As medidas mais comumente utilizadas de cobertura baseiam-se na cobertura dos requisitos de software e do código fonte. A cobertura de teste é qualquer medida de integralidade relacionada a um requisito (baseada em requisitos) ou aos critérios de design e implementação do código (baseada em códigos), como a verificação de casos de uso (baseada em requisitos) ou a execução de todas as linhas de código (baseada em códigos).

A **qualidade** é uma medida de confiabilidade, de estabilidade e de desempenho do objetivo do teste (sistema ou aplicativo em teste)**.** Ela se baseia na avaliação dos resultados do teste e na análise das solicitações de mudança (defeitos) identificadas durante o teste.

Embora a avaliação da cobertura de teste forneça uma medida da extensão da integralidade do esforço de teste, a avaliação dos defeitos descobertos durante o teste fornece a melhor indicação da qualidade de software tal como foi experienciada. Essa observação da qualidade pode ser utilizada para raciocinar sobre a qualidade geral do sistema de software como um todo.

1.2. Estratégias de teste

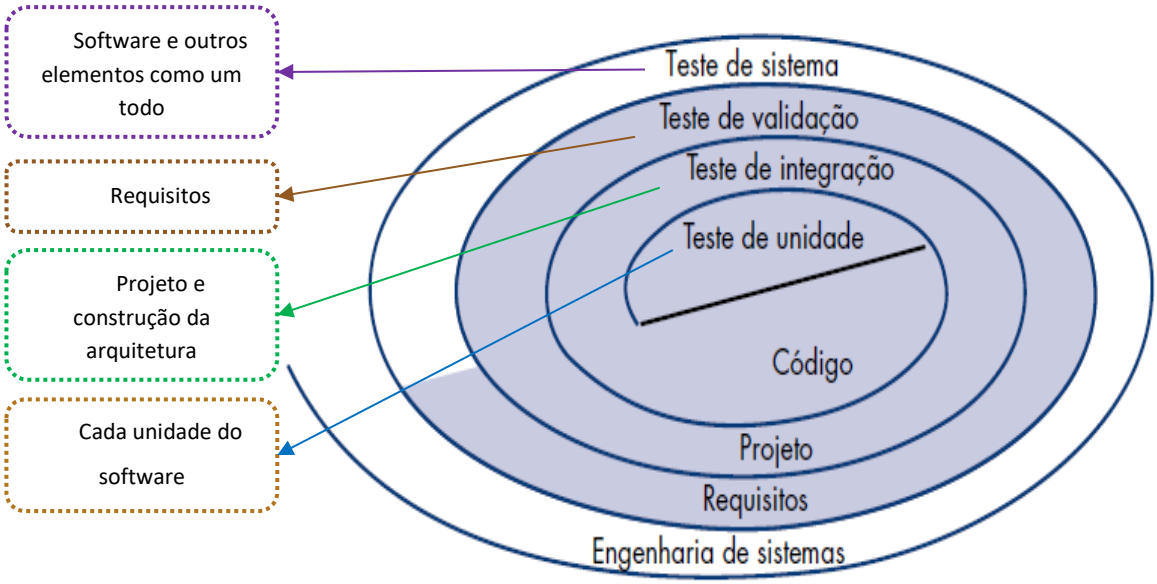
A **estratégia de teste** fornece um roteiro que descreve os passos a serem executados como parte do teste, define quando esses passos são planejados e então executados, e quanto trabalho, tempo e recursos são necessários.

Uma estratégia de teste deve ser:

- **Flexível o bastante** para promover uma abordagem de teste personalizada.
- **Rígida o bastante** para estimular um planejamento razoável e o acompanhamento, à medida que o projeto progride.
- Deve **acomodar testes de baixo nível e de alto nível**.

Pressman propõe um modelo espiral para demonstrar a estratégia de teste. O **teste de unidade** começa no centro da espiral e se concentra em cada unidade (por exemplo: componente, classe, ou objeto de conteúdo de WebApp) do software conforme implementado no código-fonte. O teste prossegue movendo-se em direção ao exterior da espiral, passando pelo **teste de integração**, em que o foco está no projeto e construção da arquitetura de software. Continuando na mesma direção da espiral, encontramos o **teste de validação**, em que requisitos estabelecidos como parte dos requisitos de modelagem são validados em relação ao software criado. Finalmente, chegamos ao **teste do sistema**, no qual o software e outros elementos são testados como um todo. Para testar um software de computador, percorre-se a espiral em direção ao seu exterior, no sentido horário, ao longo de linhas que indicam o escopo do teste a cada volta.

A figura a seguir ilustra a estragégia de teste proposta por Pressman:



1.2.1. Testes de unidade

Os **testes de unidade ou testes unitários** focalizam cada componente individualmente, garantindo que ele funcione adequadamente como uma unidade. Enfoca a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente. Usa intensamente técnicas de teste com caminhos específicos na estrutura de controle de um componente.

Normalmente, o **teste de unidade** é considerado um auxiliar na etapa de codificação. Pode ocorrer antes ou depois de começar a codificação.

Devido ao fato de um componente não ser um programa independente, deve ser desenvolvido um pseudocontrolador (driver) e/ou pseudocontrolado (stub) para cada teste de unidade.

Diferentemente do teste de unidade convencional, que tende a focalizar o detalhe algorítmico de um módulo e os dados que fluem através da interface do módulo, um **teste de unidade no contexto OO ou teste de classe** é controlado pelas operações encapsuladas na classe e pelo estado de comportamento da classe.

1.2.2. Testes de integração

No **teste de integração**, o componente é montado ou integrado para formar o pacote completo de software. Cuida de problemas associados com aspectos duais de verificação e construção do programa. É uma técnica sistemática para construir a arquitetura de software ao mesmo tempo que conduz testes para descobrir erros associados com a interface. Usa técnicas de projeto de casos de teste que focalizam em entradas e saídas. Pode usar técnicas que usam caminhos específicos de programa.

Pode ser baseado nas seguintes **abordagens convencionais**:

- **Big-Bang**: integração não incremental. Todos os componentes são combinados com antecedência. O programa é testado como um todo.
- **Incremental**: o programa é construído e integrado em pequenos incrementos.
    - **Descendente (top-down)**: os módulos são integrados deslocando-se para baixo através da hierarquia de controle, começando com o módulo de controle principal. Verifica os principais pontos de controle ou decisão antecipada no processo. Pode ser realizada: (1) primeiro-em-profundidade; (2) primeiro-em-largura.
    - **Ascendente (bottom-up)**: começa a construção e o teste com módulos atômicos.

Algumas abordagens especiais de teste de integração são:

- **Teste de regressão**: é a reexecução do mesmo subconjunto de testes que já foram executados para assegurar que as alterações não tenham propagado efeitos colaterais indesejados. Deve ser projetado de forma a incluir somente aqueles testes que tratam de uma ou mais classes de erros em cada uma das funções principais do programa. É impraticável e ineficiente reexecutar todos os testes para todas as funções do programa.
- **Teste fumaça**: séries de testes criada para expor erros que impedem uma construção (build) de executar exatamente sua função. A finalidade deverá ser descobrir erros "bloqueadores" (showstopper) que apresentam a mais alta probabilidade de atrasar o cronograma do software. A construção é integrada com outras construções, e o produto inteiro passa diariamente pelo teste fumaça.

1.2.3. Testes de validação

O **teste de validação** fornece a garantia final de que o software satisfaz aos requisitos informativos, funcionais, comportamentais e de desempenho. Focaliza ações visíveis ao usuário e saídas do sistema reconhecidas por ele.

A validação é conseguida por meio de uma série de testes que demonstram conformidade com os requisitos. Os testes de validação incluem:

- **Revisão da configuração**: visa garantir que todos os elementos da configuração do software tenham sido adequadamente desenvolvidos, estejam catalogados e tenham os detalhes necessários para amparar as atividades de suporte.

- **Teste alfa**: conduzido na instalação do desenvolvedor por um grupo representativo de usuários finais. Conduzido em um ambiente controlado, com o desenvolvedor registrando os erros e os problemas de uso.
- **Teste beta**: conduzido nas instalações de um ou mais usuários finais. O desenvolvedor geralmente não está presente. O cliente registra todos os problemas encontrados durante o teste e relata esses problemas para o desenvolvedor em intervalos regulares.
- **Teste de aceitação do cliente**: variação do teste beta. Executada quando é fornecido software personalizado a um cliente sob contrato. O cliente executa uma série de testes específicos na tentativa de descobrir erros antes de aceitar o software do desenvolvedor.

1.2.4. Testes de sistema

O **teste de sistema** verifica se os elementos se combinam corretamente e se a função/desempenho global do sistema é conseguida. Extrapola a engenharia de software e entra no contexto de engenharia de sistema.

Teste de sistema é na realidade uma série de diferentes testes cuja finalidade primária é exercitar totalmente o sistema. Embora cada um dos testes tenha uma finalidade diferente, todos funcionam no sentido de verificar se os elementos do sistema foram integrados adequadamente e executam as funções a eles alocadas.

Vejamos os principais testes de sistema:

- **Teste de recuperação:** força o software a falhar de várias formas e verifica se a recuperação é executada corretamente.
- **Teste de segurança**: tenta verificar se os mecanismos de proteção do sistema irão de fato proteger o sistema contra invasão.
- **Teste por esforço (stress)**: usa um sistema de maneira que demande recursos em quantidade, frequência ou volume anormal.
- **Teste de sensibilidade**: variação do teste stress. Tenta descobrir combinações de dados dentro de classes de entrada válidas que podem causar instabilidade ou processamento inadequado.
- **Teste de disponibilização (ou de configuração)**: exercita o software em cada ambiente no qual ele deve operar. Examina todos os procedimentos de instalação e software especializado de instalação que serão usados pelos clientes e toda a documentação que será usada para fornecer o software para os usuários finais.

1.3. Abordagens de teste

Os produtos de engenharia, incluindo os produtos de software, podem ser testados basicamente de duas maneiras:

- Conhecendo o funcionamento interno de um produto, podem ser realizados testes para garantir que "tudo se encaixa", isto é, que as operações internas foram realizadas de acordo com as especificações e que todos os componentes internos foram adequadamente exercitados. Essa abordagem requer uma visão interna e é chamada de **teste caixa-branca**.
- Conhecendo a função especificada para o qual um produto foi projetado para realizar, podem ser feitos testes que demonstram que cada uma das funções é totalmente operacional, embora ao mesmo tempo procurem erros em cada função. Esta abordagem usa uma visão externa e é chamada de **teste caixa preta**.
- É possível aplicar ambas as técnicas em conjunto, realizando-se assim os chamados **testes de caixa cinza**.

O esquema a seguir ilustra as diferenças entre essas abordagens:

1.3.1. Testes de caixa branca

O **teste de caixa branca** é um teste projetado para usar a estrutura de controle descrita como parte do projeto de componentes para derivar os casos de teste. O engenheiro pode criar casos de teste que:

- Garantam que **todos os caminhos independentes** de um módulo foram exercitados pelo menos uma vez.
- Exercitam **todas as decisões lógicas** nos seus estados verdadeiro e falso.
- Executam **todos os ciclos** em seus limites e dentro de suas fronteiras operacionais.
- Exercitam **estruturas de dados** internas para assegurar a validade.

Os testes de caixa branca podem ser realizados com base nos seguintes **testes de estruturas de controle**:

- **Teste de caminho básico**: permite ao projetista de casos de teste derivar uma medida da complexidade lógica de um projeto procedimental e usar essa medida como guia para definir um conjunto base de caminhos de execução. Casos de teste criados para exercitar o conjunto básico executam com certeza todas as instruções de um programa pelo menos uma vez durante o teste.
- **Teste de condição**: método de projeto de caso de teste que exercita as condições lógicas contidas em um módulo de programa.
- **Teste de fluxo de dados**: seleciona caminhos de teste de um programa de acordo com as localizações de definições e usos de variáveis no programa.
- **Teste de ciclo (loop)**: focaliza exclusivamente a validade das construções de ciclo.

1.3.2. Testes de caixa preta

O **teste de caixa preta** focaliza os requisitos funcionais do software. O teste caixa-preta não é uma alternativa às técnicas caixa-branca. Em vez disso, é uma abordagem complementar, com possibilidade de descobrir uma classe de erros diferente daquela obtida com métodos caixa-branca.

O **teste caixa preta** tenta encontrar erros nas seguintes categorias:

- Funções incorretas ou faltando.
- Erros de interface.
- Erros em estruturas de dados ou acesso a bases de dados externas.
- Erros de comportamento ou de desempenho.
- Erros de inicialização e término.

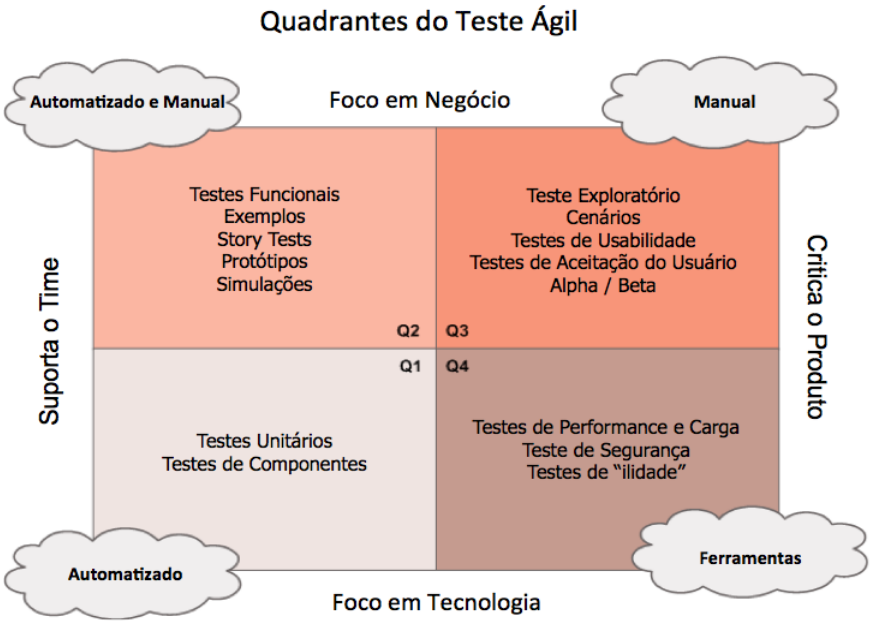
Testes de caixa preta podem ser realizados com os seguintes métodos:

- **Métodos de teste com base em grafo**: começa criando um grafo de objetos importantes e suas relações e então imaginando uma série de testes que abrangerá o grafo de forma que cada objeto e relação sejam exercitados e os erros sejam descobertos.
- **Particionamento de equivalência**: divide o domínio de entrada de um programa em classes de dados a partir das quais podem ser criados casos de teste.
- **Análise de valor limite**: leva a uma seleção de casos de teste que utilizam valores limites. Contempla o particionamento de equivalência.

- **Teste de matriz ortogonal**: forma sistemática, estatística de teste de software. É usado quando o número de entradas para o sistema é relativamente pequena, mas grande demais para permitir testes exaustivos de cada entrada possível para os sistemas. O objetivo do teste é a construção de caso de teste com uma visualização geométrica associada aos valores de entrada de uma aplicação.
- **Teste baseado em modelos**: usa informações contidas no modelo de requisitos como base para a geração de casos de teste.

1.4. Testes em ambientes ágeis

O **Quadrante de Teste Ágil** foi criado por Brian Marick que introduziu uma série de termos para diferentes categorias de teste para alcançar diferentes propósitos através de um diagrama.



1.4.1. Quadrante 1

O Quadrante 1 representa, basicamente, a principal prática de desenvolvimento ágil: TDD – Test Driven Development.

Dentro deste quadrante temos duas práticas: teste de unidade, que valida uma pequena parte da aplicação como objetos e/ou métodos, e testes de componente que valida partes maiores da aplicação como um grupo de classes que provê o mesmo serviço. São usualmente desenvolvidos com ferramentas xUnit e medem a qualidade interna do produto.

Não são destinados ao cliente, pois o cliente não irá entender os aspectos internos do desenvolvimento e não devem ser negociáveis com o cliente, pois a qualidade do código deve sempre existir e não ser negligenciada. Os testes desenvolvidos usualmente executados dentro de uma abordagem de Integração Contínua para prover um rápido feedback da qualidade do código.

1.4.2. Quadrante 2

Este quadrante também suporta o time de desenvolvimento, continua guiando o desenvolvimento, mas de uma maneira mais alto-nível focando mais em testes que o cliente entenda, onde este define a qualidade externa de que ele precisa.

Aqui o cliente define exemplos que serão usados para um entendimento maior do funcionamento da aplicação, e são escritos de forma com que o cliente ou papéis ligados a negócio entendam. Todo o teste executado aqui tem um foco no funcionamento do produto e alguns deles podem até mesmo ter uma pequena duplicação com alguns testes criados no Quadrante 1.

Testes focados no negócio também podem ser automatizados e, usualmente, a técnica de BDD – Behavior Driven Development é utilizada na escrita e execução automatizada destes exemplos.

Neste quadrante também podemos utilizar de pessoas com conhecimentos em UX (User Experience) para que, através de mockups e wireframes, o cliente possa validar a interface gráfica antes que o time comece a desenvolver esta camada.

1.4.3. Quadrante 3

As vezes mesmo criando diversos mecanismos para assegurar que estamos atendendo a necessidade do cliente através de critérios e/ou exemplos, pode acontecer de não atendermos realmente aquele desejo, ou mesmo o teste unitário e o teste através do BDD podem não ter um real valor.

Neste quadrante iremos realmente criticar o produto e executá-lo como um usuário real usando nosso conhecimento e intuição na utilização da aplicação. O cliente pode executar este tipo de tarefa, usualmente chamada de UAT – User Acceptance Testing (teste de aceitação), dando um feedback mais preciso, aceitando a funcionalidade, analisando possíveis novas funcionalidades.

O ponto central deste quadrante, além do UAT, são os testes exploratórios. Utilizando esta técnica qualquer membro do time é capaz de, simultaneamente, aprender sobre a aplicação e executar mais testes, usando o feedback do último teste para a execução dos próximos e também é capaz de extrair novos critérios, sempre observando o comportamento da aplicação.

1.4.4. Quadrante 4

Os testes neste quadrante são os mais técnicos e criticam o produto em termos de performance, carga e segurança.

Nos dias de hoje negligenciar aspectos como performance podem tirar a vantagem competitiva de um cliente. As técnicas aplicadas a performance, carga e segurança vão desde os níveis mais baixos como a utilização de ferramentas que simulam diversos usuários simultaneamente.

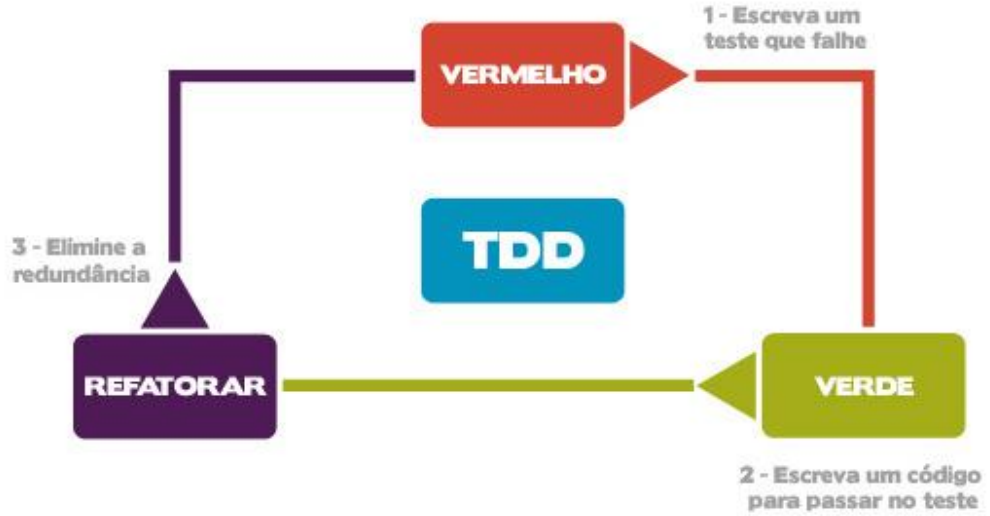
1.5. Test Driven Development (TDD)

O **TDD (Test Driven Development) ou Desenvolvimento Orientado a Testes** é uma técnica ou filosofia de programação que incorpora o teste ao processo de produção de código da seguinte forma:

- Primeiramente o desenvolvedor, que recebe a especificação de uma nova funcionalidade a ser implementada, deve **programar um conjunto de testes** automatizados para testar o código que ainda não existe.
- Esse conjunto de testes deve ser **executado e falhar**. Isso é feito para mostrar que os testes efetivamente têm algum poder de teste e não terão sucesso a não ser que o código específico seja desenvolvido. Se o código passar em algum desses testes, é possível que ou o teste não seja efetivo, ou que a característica a ser implementada já exista no sistema.
- Em seguida, o **código deve ser desenvolvido** da forma mais minimalista possível, isto é, apenas para passar nos testes.
- Depois que o código **passar em todos os testes**, deve **ser refatorado** para atender a padrões de qualidade interna e testado novamente até que passe em todos os testes novamente.

O ciclo do TDD é mostrado na imagem abaixo

Simplificadamente, são usadas três cores para representar as fases do ciclo TDD:



Os testes, inclusive no TDD, devem ser:

- **F (Fast) - Rápidos**: devem ser rápidos, pois testam apenas uma unidade;
- **I (Isolated) - Isolados**: Testes unitários são isolados, testando individualmente as unidades e não sua integração;
- **R (Repeateble) – Repetíveis:** Repetição nos testes, com resultados de comportamento constante;
- **S (Self-verifying) – Auto-verificáveis:** A auto verificação deve verificar se passou ou se deu como falha o teste;
- **T (Timely)** – **Tempestivos**: o teste deve ser oportuno, sendo um teste por unidade.

REFERÊNCIAS

ADAPTWORKS. **Introdução do quadrante de teste ágil**. Disponível em: <http://blog.adaptworks.com.br/2013/11/introducao-do-quadrante-de-teste-agil/>. Acesso em: 09 jan. 2019.

DEVMEDIA. **TDD: fundamentos do desenvolvimento orientado a testes**. Disponível em: <https://www.devmedia.com.br/tdd-fundamentos-do-desenvolvimento-orientado-a-testes/28151>. Acesso em: 09 jan. 2019.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

PRESSMAN, Roger. **Engenharia de Software**. 7. ed. Porto Alegre: AMGH, 2011.

2. UNIDADE 2 – ISO/IEC 9126

2.1. Introdução a ISO/IEC 9126

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

Neste contexto, surge a NBR ISO/IEC 9126, sob o título geral "Engenharia de Software – Qualidade do Produto". Esta norma é subdivida nas seguintes partes:

- **Parte 1**: Modelo de qualidade;
- **Parte 2**: Métricas externas;
- **Parte 3**: Métricas internas;
- **Parte 4**: Métricas de qualidade de uso.

Vamos tratar do modelo de qualidade descrito na Parte 1 da norma, também referido como NBR ISO/IEC 9126-1.

O modelo de qualidade especifica seis características para a qualidade interna e externa. Essas características são subdivididas em subcaracterísticas, que são manifestadas externamente quando o software é utilizado como parte de um sistema computacional e são resultantes de atributos internos do software.

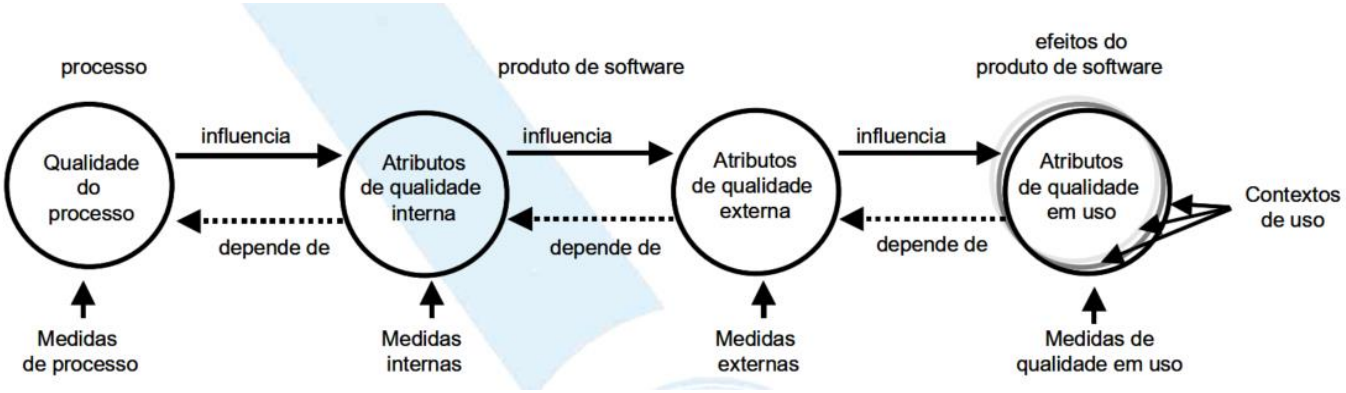
Além disso, o modelo apresenta as quatro características da qualidade em uso, que é, para o usuário, o efeito combinado das características de qualidade do produto.

2.2. Estrutura do modelo de qualidade

2.2.1. Abordagens de qualidade

A NBR ISO/IEC 9126-1 descreve uma estrutura que explica o relacionamento entre diferentes abordagens de qualidade.

As abordagens são ilustradas na figura a seguir:



As necessidades de qualidade do usuário incluem requisitos de qualidade em uso em contextos de uso específicos. Estas necessidades identificadas podem ser usadas na especificação da qualidade interna e externa, aplicando características e subcaracterísticas de qualidade do produto de software.

A qualidade do produto de software pode ser avaliada medindo-se os atributos internos (tipicamente medidas estáticas de produtos intermediários), os atributos externos (tipicamente pela medição do comportamento do código quando executado) ou os atributos de qualidade em uso.

A qualidade de processo contribui para melhorar a qualidade do produto e a qualidade do produto contribui para melhorar a qualidade em uso. Por isso, avaliar e melhorar o processo é um meio de melhorar a qualidade do produto, assim como avaliar e melhorar a qualidade do produto é um meio de melhorar a qualidade em uso.

2.2.2. Requisitos de qualidade

As **necessidades de qualidade do usuário** podem ser especificadas como requisitos de qualidade pelas métricas de qualidade em uso, pelas métricas externas e algumas vezes por métricas internas.

Os **requisitos de qualidade externa** especificam o nível de qualidade requerido sob o ponto de vista externo. Incluem requisitos derivados das necessidades de qualidade dos usuários, incluindo os requisitos de qualidade em uso.

A **qualidade externa** é a totalidade das características do produto de software do ponto de vista externo. É a qualidade quando o software é executado, o qual é tipicamente medido e avaliado enquanto está sendo testado num ambiente simulado, com dados simulados e usando métricas externas.

Os **requisitos de qualidade interna** especificam o nível de qualidade requerido sob o ponto de vista interno do produto. Os requisitos de qualidade interna são usados para especificar as propriedades dos produtos intermediários. Estes podem incluir modelos estáticos e dinâmicos, outros documentos e código-fonte.

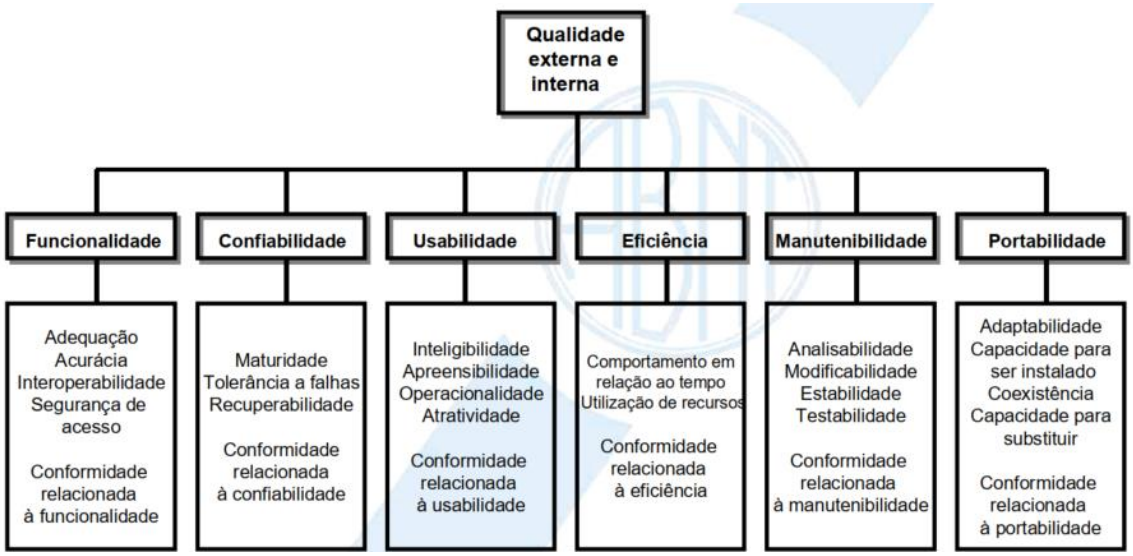
A **qualidade interna** é a totalidade das características do produto de software do ponto de vista interno. A qualidade interna é medida e avaliada com relação aos requisitos de qualidade interna. Detalhes da qualidade do produto de software podem ser melhorados durante a implementação do código, revisão e teste, mas a natureza fundamental da qualidade do produto de software representada pela qualidade interna mantém-se inalterada, a menos que seja reprojetada.

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

2.3. Modelo de qualidade para qualidade externa e interna

O **modelo de qualidade externa e interna** categoriza os atributos de qualidade de software em seis características (funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade) as quais são, por sua vez, subdivididas em subcaracterísticas. As subcaracterísticas podem ser medidas por meio de métricas externas e internas.

A figura a seguir mostra as características e subcaracterísticas:



Vamos descrever cada uma dessas características:

2.3.1. Funcionalidade

A **funcionalidade** é a capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas. Esta característica está relacionada com o que software faz para atender às necessidades.

A funcionalidade contém as seguintes subcaracterísticas:

- **Adequação**: capacidade do produto de software de prover um conjunto apropriado de funções para tarefas e objetivos do usuário especificados.
- **Acurácia**: capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados.
- **Interoperabilidade**: capacidade do produto de software de interagir com um ou mais sistemas especificados.
- **Segurança de acesso**: capacidade do produto de software de proteger informações e dados, de forma que pessoas ou sistemas não autorizados não possam lê-los nem os modificar e que não seja negado o acesso às pessoas ou sistemas autorizados.
- **Conformidade relacionada à funcionalidade**: capacidade do produto de software de estar de acordo com normas, convenções ou regulamentações previstas em leis e prescrições similares relacionadas à funcionalidade.

2.3.2. Confiabilidade

A **confiabilidade** é a capacidade do produto de software de manter um nível de desempenho especificado, quando usado em condições especificadas.

Em software não ocorre desgaste ou envelhecimento. As limitações em confiabilidade são decorrentes de defeitos na especificação de requisitos, projeto e implementação. As falhas decorrentes desses defeitos dependem de como o produto de software é usado e das opções de programa selecionadas e não do tempo decorrido.

A confiabilidade contém as seguintes subcaracterísticas:

- **Maturidade**: capacidade do produto de software de evitar falhas decorrentes de defeitos no software.
- **Tolerância a falhas**: capacidade do produto de software de manter um nível de desempenho especificado em casos de defeitos no software ou de violação de sua interface especificada.

- **Recuperabilidade**: capacidade do produto de software de restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha.
- **Conformidade relacionada à confiabilidade**: capacidade do produto de software de estar de acordo com normas, convenções ou regulamentações relacionadas à confiabilidade.

2.3.3. Usabilidade

A **usabilidade** é a capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas.

A usabilidade contém as seguintes subcaracterísticas:

- **Inteligibilidade**: capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas.
- **Apreensibilidade**: capacidade do produto de software de possibilitar ao usuário aprender sua aplicação.
- **Operacionalidade**: capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo.
- **Atratividade**: capacidade do produto de software de ser atraente ao usuário. Isto refere-se a atributos de software que possuem a intenção de tornar o software mais atraente para o usuário, como o uso de cores e da natureza do projeto gráfico.
- **Conformidade relacionada à usabilidade**: capacidade do produto de software de estar de acordo com normas, convenções, guias de estilo ou regulamentações relacionadas à usabilidade.

2.3.4. Eficiência

A **eficiência** é a capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas.

A eficiência contém as seguintes subcaracterísticas:

- **Comportamento em relação ao tempo**: capacidade do produto de software de fornecer tempos de resposta e de processamento, além de taxas de transferência, apropriados, quando o software executa suas funções, sob condições estabelecidas.
- **Utilização de recursos**: capacidade do produto de software de usar tipos e quantidades apropriados de recursos, quando o software executa suas funções sob condições estabelecidas.
- **Conformidade relacionada à eficiência**: capacidade de estar de acordo com normas e convenções relacionadas à eficiência.

2.3.5. Manutenibilidade

A **manutenibilidade** é a capacidade do produto de software de ser modificado. As modificações podem incluir correções, melhorias ou adaptações.

A manutenibilidade contém as seguintes subcaracterísticas:

- **Analisabilidade**: capacidade do produto de software de permitir o diagnóstico de deficiências ou causas de falhas no software, ou a identificação de partes a serem modificadas.
- **Modificabilidade**: capacidade do produto de software de permitir que uma modificação especificada seja implementada.
- **Estabilidade**: capacidade do produto de software de evitar efeitos inesperados decorrentes de modificações no software.
- **Testabilidade**: capacidade do produto de software de permitir que o software, quando modificado, seja validado.
- **Conformidade relacionada à manutenibilidade**: capacidade do produto de software de estar de acordo com normas ou convenções relacionadas à manutenibilidade.

2.3.6. Portabilidade

A **portabilidade** é a capacidade do produto de software de ser transferido de um ambiente para outro. O ambiente pode ser organizacional, de hardware ou de software.

A portabilidade contém as seguintes subcaracterísticas:

- **Adaptabilidade**: capacidade do produto de software de ser adaptado para diferentes ambientes especificados, sem necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado.
- **Capacidade para ser instalado**: capacidade do produto de software para ser instalado em um ambiente especificado.
- **Coexistência**: capacidade do produto de software de coexistir com outros produtos de software independentes, em um ambiente comum, compartilhando recursos comuns.
- **Capacidade para substituir**: capacidade do produto de software de ser usado em substituição a outro produto de software especificado, com o mesmo propósito e no mesmo ambiente.
- **Conformidade relacionada à portabilidade**: capacidade do produto de software de estar de acordo com normas ou convenções relacionadas à portabilidade.

2.4. Modelo de qualidade para qualidade em uso

O **modelo de qualidade para qualidade em uso** categoriza os atributos de qualidade em quatro características: eficácia, produtividade, segurança e satisfação.

Vejamos essas características:

- **Eficácia**: capacidade do produto de software de permitir que usuários atinjam metas especificadas com acurácia e completitude, em um contexto de uso especificado.
- **Produtividade**: capacidade do produto de software de permitir que seus usuários empreguem quantidade apropriada de recursos em relação à eficácia obtida, em um contexto de uso especificado.
- **Segurança**: capacidade do produto de software de apresentar níveis aceitáveis de riscos de danos a pessoas, negócios, software, propriedades ou ao ambiente, em um contexto de uso especificado.
- **Satisfação**: capacidade do produto de software de satisfazer usuários, em um contexto de uso especificado. Satisfação é a resposta do usuário à interação com o produto e inclui atitudes relacionadas ao uso do produto.

REFERÊNCIAS

NBR. **NBR ISO/IEC 9126-1**. Engenharia de software – Qualidade de Software. Parte 1: Modelo de Qualidade. Rio de Janeiro, 2003.

3. UNIDADE 3 – ANÁLISE DE PONTO DE FUNÇÃO

3.1. Introdução a Análise de Pontos de Função

A utilização de pontos de função como medida do tamanho funcional do software tem crescido desde os meados da década de 70, de umas poucas organizações interessadas até uma impressionante lista de organizações no mundo inteiro.

O crescimento da utilização de pontos de função tem ampliado a aplicação e utilização da medida. Desde sua fundação em 1986, o International Function Point Users Group (IFPUG) tem aprimorado continuamente o método para o dimensionamento funcional de software.

A **análise de pontos de função (APF)** mede o software quantificando as tarefas e serviços (isto é, funcionalidade) que o software fornece ao usuário, primordialmente com base no projeto lógico.

Os **objetivos da análise de pontos de função** são medir:

- A funcionalidade implementada no software, que o usuário solicita e recebe;
- A funcionalidade impactada pelo desenvolvimento, melhoria e manutenção de software, independentemente da tecnologia utilizada na implementação.

O **processo** de APF é:

- Suficientemente simples para minimizar o custo adicional introduzido pelo processo de medição;
- Uma medida consistente entre diversos projetos e organizações.

O método pode medir **apenas "tamanho funcional"** e **não "tamanho não-funcional"**.

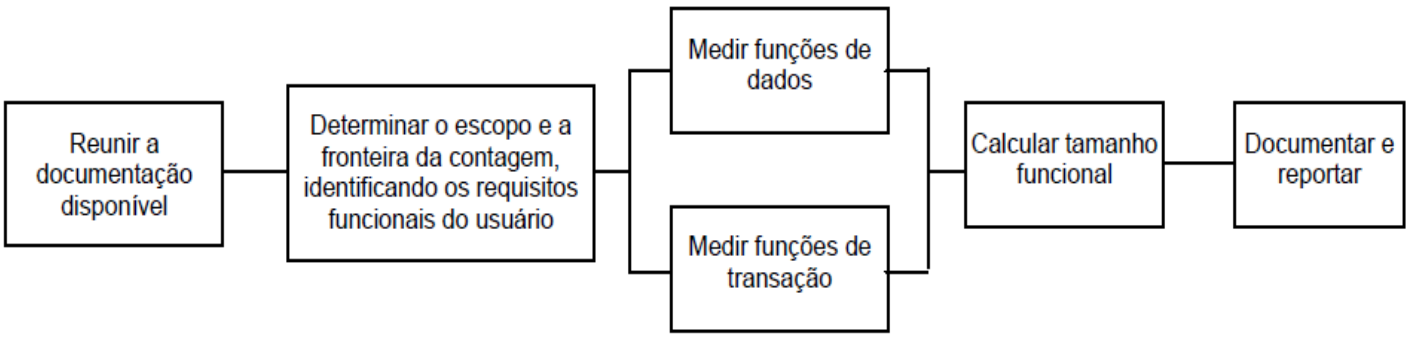
- **Requisito Funcional**: um subconjunto dos requisitos do usuário que descrevem o que o software deve fazer, em termos de tarefas e serviços.
- **Requisito Não Funcional**: restrições de qualidade, organizacionais, ambientais e de implementação.

O **padrão do IFPUG** especifica o conjunto de definições, regras e passos para a aplicação do método de medição de tamanho funcional (FSM) do IFPUG.

Os analistas de pontos de função do IFPUG identificaram diferentes taxas de entrega (horas para entregar um ponto de função) relacionadas à construção de aplicações em diferentes domínios funcionais, calibradas para diversos tamanhos de projeto e complexidade do software.

3.2. Processo de Medição

Para conduzir uma contagem de pontos de função devem ser executadas as seguintes atividades, a fim de identificar e classificar os componentes funcionais básicos:



3.2.1. Reunir a documentação disponível

A **documentação** de suporte a uma contagem de PF deve descrever a funcionalidade entregue pelo software ou a funcionalidade impactada pelo projeto de software medido.

Deve ser obtida documentação suficiente para conduzir a contagem de pontos de função, ou acesso a especialistas no assunto capazes de fornecer informações adicionais para suprir quaisquer falhas na documentação.

É importante considerar que a APF não é estritamente técnica e, portanto, deve considerar as funcionalidades conforme a visão do usuário.

A **visão do usuário** representa uma descrição formal das necessidades dos negócios do usuário, na linguagem do usuário. Os desenvolvedores traduzem a informação do usuário para informações em linguagem técnica a fim de prover uma solução.

Uma **medição** de tamanho funcional é realizada utilizando a informação em uma linguagem que é comum para o usuário(s) e desenvolvedores.

Os requisitos evoluem no processo gerando basicamente as seguintes documentações:

- **Requisitos Iniciais do Usuário**: representa os requisitos dos usuários antes das sessões entre os usuários e os desenvolvedores de software. Podem ser: incompletos; faltar funcionalidades "utilitárias"; impossibilidade de implementação ou uso muito difícil; muito genéricos; não atender as necessidades para todos os usuários; não considerar as fronteiras da aplicação; expressos em um contexto diferente ou terminologia não compatível com APF.

- **Requisitos Técnicos Iniciais**: representa a visão dos desenvolvedores de software sobre os requisitos criados a partir do estudo de viabilidade. Podem incluir elementos necessários para a implementação, mas não são utilizados na medição de tamanho funcional. Apresentam como características: dependência tecnológica; terminologia não familiar com os usuários; funcionalidades podem ser determinadas enfatizando restrições técnicas; fronteiras são determinadas de acordo com a arquitetura técnica ao invés de processos de negócio.
- **Requisitos Funcionais Finais**: origina-se de sessões conjuntas entre o(s) usuário(s) e o(s) desenvolvedor(es). Tem as seguintes características: terminologia entendida pelos usuários e desenvolvedores; descrições integradas de todos os requisitos; todos os processos de negócio completamente definidos; cada processo e grupo de dados é aprovado por usuário e desenvolvedor; a viabilidade e utilidade são aprovadas pelos desenvolvedores.

Antes de iniciar uma medição de tamanho funcional, determine a fase do ciclo de vida da aplicação e se você vai fazer uma aproximação, ou uma medição.

- **Aproximação**: permite fazer suposições sobre funções desconhecidas e/ou suas complexidades, para determinar um tamanho funcional aproximado.
- **Medição**: inclui a identificação de todas as funções e suas complexidades, para efetuar uma análise de pontos de função.

Num primeiro estágio, os Requisitos Iniciais do Usuário podem ser o único documento disponível para a análise de pontos de função. Apesar das desvantagens, este tamanho pode ser muito útil para produzir uma estimativa antecipada.

3.2.2. Determinar o escopo e a fronteira da contagem

Para **determinar o escopo e fronteira da contagem e identificar os Requisitos Funcionais do Usuário**, devem ser executadas as seguintes atividades:

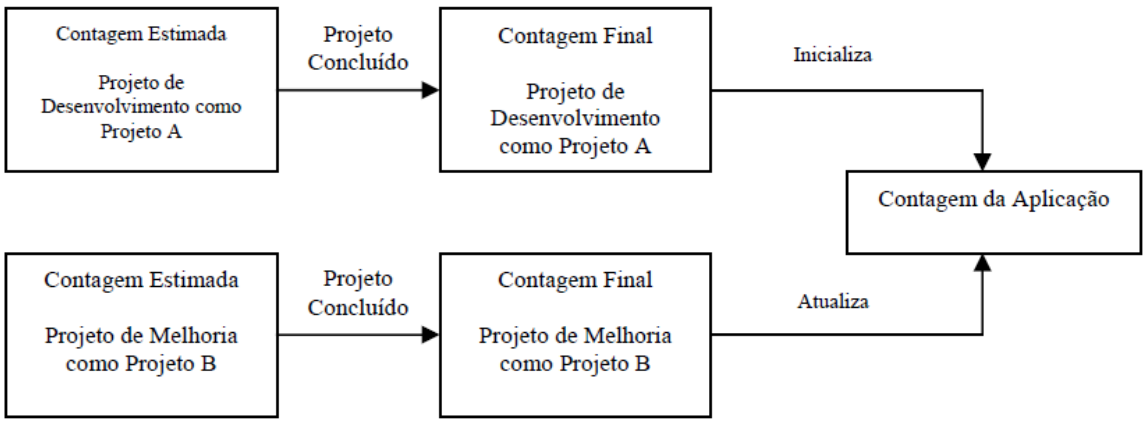
1. **Identificar o propósito da contagem**: o propósito determina o tipo de contagem de pontos de função e o escopo da contagem necessária para obter a resposta ao problema de negócios sob investigação e influencia o posicionamento da fronteira entre o software sob análise e o software vizinho.
2. **Identificar o tipo de contagem**: o tipo de contagem é definido com base no propósito e pode ser de:
    a. **Projeto de desenvolvimento**: Um projeto de desenvolvimento é um projeto para desenvolver e fornecer a primeira versão de um software. O tamanho funcional do projeto de desenvolvimento é uma medida de funcionalidade oferecida aos usuários com a primeira

instalação do software, conforme medido pela contagem de pontos de função do projeto de desenvolvimento pela atividade de aplicação, o método de medição funcional (FSM) IFPUG.

b. **Projeto de melhoria**: um projeto de melhoria é um projeto para desenvolver e entregar manutenção adaptativa. O tamanho funcional do projeto de melhoria é uma medida das funcionalidades adicionadas, alteradas e excluídas na conclusão de um projeto de melhoria, conforme medido pela contagem dos pontos de função do projeto de melhoria pela atividade de aplicação do método de Medição de Tamanho Funcional (FSM) do IFPUG.

c. **Aplicação**: uma aplicação é uma coleção coesa de procedimentos automatizados e dados apoiando um objetivo de negócio; isto consiste em um ou mais componentes, módulos ou subsistemas. Um tamanho funcional de uma aplicação é uma medida de funcionalidade que uma aplicação oferece ao usuário, determinado pela contagem de pontos de função da aplicação pela atividade de aplicação do método de Medição de Tamanho Funcional (FSM) do IFPUG. Ela também e chamado de baseline ou tamanho funcional instalado. Este tamanho fornece uma medida de funções atuais que o aplicativo fornece ao usuário. O número é inicializado quando o projeto de desenvolvimento da contagem de pontos de função é finalizado. É atualizado toda vez que um projeto de melhoria finalizado alterar funções da aplicação.

O diagrama a seguir ilustra o relacionamento entre os tipos de contagem:



3. **Determinar o escopo da contagem**: o escopo da contagem define o conjunto de Requisitos Funcionais de Usuários para ser incluído na contagem de pontos de função.

O **escopo**:

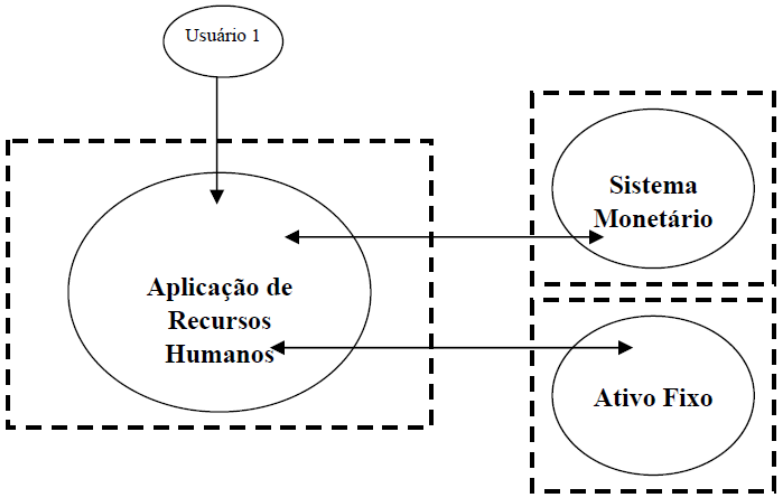
a. Define o (sub)conjunto do software que está sendo medido.

b. É determinado pelo propósito para a realização da contagem de pontos de função.

c. Identifica quais funções serão incluídas na medida de tamanho funcional assim como fornecer respostas relevantes para o propósito da contagem.

d. Pode incluir mais de uma aplicação.

4. **Determinar a fronteira de cada aplicação contida no escopo**: A **fronteira** é uma interface conceitual entre o software sob estudo e seus usuários. É importante destacar que a fronteira é definida com base na visão do usuário.



5. **Identificar requisitos funcionais** e excluir os não funcionais.

3.2.3. Medir funções de dados

As **funções de dados** representam a funcionalidade oferecida ao usuário para satisfazer requisitos de dados internos e externos. Uma função de dado pode ser um **arquivo lógico interno (ALI)** ou um **arquivo de interface externo (AIE)**.

O termo **arquivo** aqui não significa arquivo físico ou tabela. Nesse caso, **arquivo** se refere a um grupo de dados logicamente relacionados e não à implementação física destes grupos de dados.

Vejamos o que são arquivos lógicos internos e arquivos de interfaces externas:

- Um **arquivo lógico interno (ALI)** é um grupo de dados ou de informações de controle logicamente relacionados, reconhecido pelo usuário, mantido dentro da fronteira da aplicação que está sendo contada. A intenção primária de um ALI é armazenar dados mantidos através de um ou mais processos elementares da aplicação que está sendo contada.

- Um **arquivo de interface externa (AIE)** é um grupo de dados ou de informações de controle logicamente relacionados, reconhecido pelo usuário, referenciado pela aplicação que está sendo contada, porém, mantido dentro da fronteira de uma outra aplicação. A intenção primária de um AIE é armazenar dados referenciados através de um ou mais processos elementares dentro da fronteira da aplicação que está sendo contada. Isto significa que um AIE contado para uma aplicação deve ser um ALI em outra aplicação.

Vejamos as definições de alguns termos que estão presentes nos conceitos de ALI e AIE:

- **Informações de controle** são dados que influenciam um processo elementar da aplicação que está sendo contada. Especificam o que, quando, ou como dados serão processados.
- O termo **reconhecido pelo usuário** refere-se a requisitos definidos para processos e/ou grupos de dados que foram acordados e entendidos tanto pelos usuários quanto pelos desenvolvedores de software.
- O termo **mantido** é a capacidade de modificar dados através de um processo elementar.
- Um **processo elementar** é a menor unidade de atividade que tem significado para o(s) usuário(s). O processo elementar deve ser autocontido e deixar o negócio da aplicação que está sendo contada em um estado consistente.

Para **identificar funções de dados**, as atividades a seguir devem ser cumpridas:

- **Identificar** no escopo da contagem **todos os dados e informações de controle** logicamente relacionados e reconhecidos pelo usuário.
- **Excluir entidades que não são mantidas** por nenhuma aplicação.
- **Agrupar entidades relacionadas** que são dependentes.

Nota: Entidades independentes devem ser consideradas grupos lógicos de dados separados.

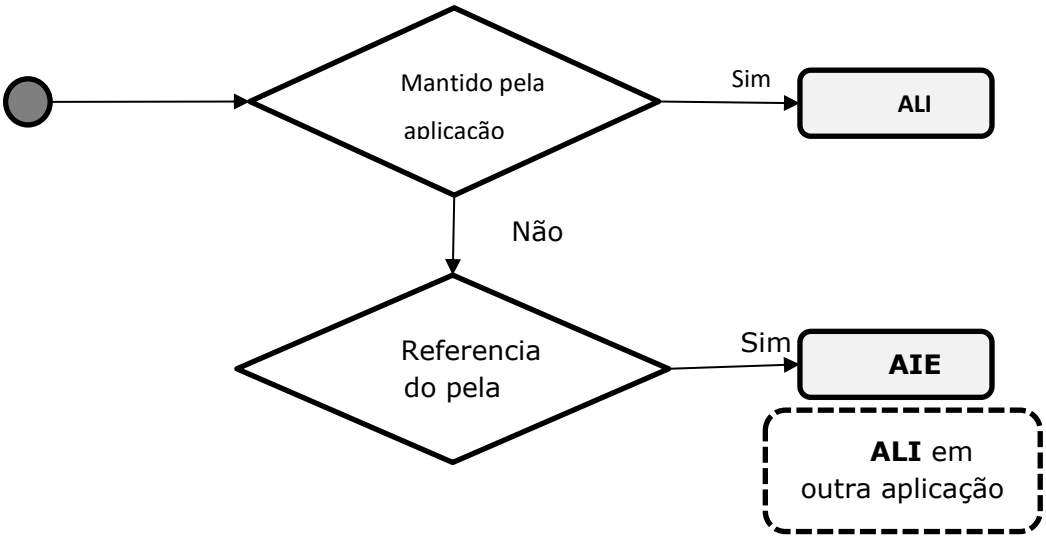
- **Excluir as entidades classificadas como Dados de código**.
- **Excluir entidades que não contém atributos necessários** para o usuário.
- **Remover entidades associativas que contém atributos adicionais não necessários para o usuário** e entidades associativas que contém apenas chaves estrangeiras; agrupe atributos de chave estrangeira com as entidades primárias.

Nota: Atributos chave estrangeira são dados necessários para o usuário estabelecer uma relação com outra função de dado.

Para **classificar uma função de dados** como ALI ou AIE:

- Classificar como um **ALI** se o dado é mantido pela aplicação que está sendo medida.

- Classificar como um **AIE** se:
  - É referenciado, mas não mantido, pela aplicação que está sendo medida e
  - É identificado como um ALI em uma ou mais aplicações**.**



O número de ALIs, AIEs, e suas respectivas complexidades funcionais determinam a contribuição das funções de dados para o tamanho funcional.

A **complexidade funcional** das funções de dados é medida com base na quantidade de tipos de dados elementares (DERs) e de tipos de registros elementares (RLRs) associados ao ALI ou AIE.

- Um **tipo de dado elementar (DER)** é um campo único, reconhecido pelo usuário e não repetido.

- Um **Tipo de Registro Elementar (RLR)** é um subgrupo de dados reconhecido pelo usuário dentro de uma função de dados.

A complexidade funcional de cada função de dados deve ser determinada utilizando o número de RLRs e DERs de acordo com a matriz seguinte:

	1 a 19 DERs	20 a 50 DERs	51 ou mais DERs
1 RLR	Baixa	Baixa	Média
2 a 5 RLRs	Baixa	Média	Alta
6 ou mais RLRs	Média	Alta	Alta

O tamanho funcional de cada função de dados é determinado usando o tipo e a complexidade funcional de acordo com a tabela a seguir:

Tamanho das funções de dados

		Tipo	
		ALI	AIE
Complexidade funcional	Baixa	7	5
	Média	10	7
	Alta	15	10

EXEMPLO

Para calcular o tamanho funcional das funções de dados, basta multiplicar o número de funções de cada tipo, pelos seus respectivos tamanhos funcionais. Vejamos um exemplo de uma aplicação que possua:

1 ALI de complexidade alta.

2 ALIs de complexidade média.

5 ALIs de complexidade baixa.

2 AIEs de complexidade alta.

5 AIEs de complexidade média.

1 AIEs de complexidade baixa.

Logo, para chegar no tamanho funcional:

1 ALI de complexidade alta = 1 x 15 = 15

2 ALIs de complexidade média= 2 x 10 = 20

5 ALIs de complexidade baixa= 5 x 7 = 35

2 AIEs de complexidade alta= 2 x 10 = 20

5 AIEs de complexidade média= 5 x 7 = 35

1 AIEs de complexidade baixa= 1 x 5 = 05

TOTAL = 130

Então, o tamanho do software em questão, considerando apenas as funções de dados, é de **130 pontos de função**.

3.2.4. Medir funções de transação

Uma **função de transação** é um processo elementar que oferece funcionalidade ao usuário para processar dados. Uma função de transação é uma entrada externa, saída externa, ou consulta externa:

- Uma **entrada externa (EE)** é um processo elementar que processa dados ou informações de controle que vêm de fora da fronteira da aplicação. A intenção primária de uma EE é manter um ou mais ALIs e/ou alterar o comportamento do sistema.
- Uma **saída externa (SE)** é um processo elementar que envia dados ou informações de controle para fora da fronteira da aplicação. A intenção primária de uma SE é apresentar informações ao usuário através de lógica de processamento que pode incluir, ou não, a recuperação de dados ou informações de controle. O processamento lógico deve conter pelo menos uma fórmula matemática ou cálculo, criar dados derivados, manter um ou mais ALIs ou alterar o comportamento do sistema.
- Uma **consulta externa (CE)** é um processo elementar que envia dados ou informações de controle para fora da fronteira da aplicação. A intenção primária de uma CE é apresentar informações ao usuário através da recuperação de dados ou informações de controle de um ALI ou AIE. O processamento lógico não deve conter fórmulas matemáticas ou cálculos, nem criar dados derivados. Nenhum ALI é mantido durante o processamento e nem o comportamento do sistema é alterado.

As funções de transação envolvem a lógica de processamento.

A **Lógica de Processamento** é definida como qualquer um dos requisitos especificamente solicitados pelo usuário para completar um processo elementar como validações, algoritmos ou cálculos e leitura ou manutenção de uma função de dados.

Formas de lógica de processamento	Tipo da Função de Transação		
	EE	SE	CE
1. Validações são efetuadas	p	p	p
2. Cálculos matemáticos são efetuados	p	d*	n
3. Valores equivalentes são convertidos	p	p	p
4. Dados são filtrados e selecionados por critérios específicos para comparar vários grupos de dados	p	p	p
5. Condições são analisadas para determinar quais se aplicam	p	p	p
6. Pelo menos um ALI é atualizado	d*	d*	n
7. Pelo menos uma ALI ou AIE é referenciado	p	p	d
8. Dados ou informações de controle são recuperados	p	p	d
9. Dados derivados são criados	p	d*	n
10. O comportamento do sistema é alterado	d*	d*	n
11. Preparar e apresentar informações para fora da fronteira	p	d	d
12. Dados ou informações de controle entrando pela fronteira da aplicação são aceitos	d	p	p
13. Os dados são reclassificados ou reorganizados	p	p	p

Legenda:
- **d** o tipo de função deve executar esta forma de lógica de processamento
- **d*** o tipo de função deve executar pelo menos uma destas formas de lógica de processamento
- **p** o tipo de função pode executar esta forma de lógica de processamento, mas a mesma não é obrigatória
- **n** o tipo de função não pode executar esta forma de lógica de processamento

Assim, note que, por exemplo, as consultas externas não podem efetuar cálculos matemáticos, atualizar ALIs, criar derivações ou alterar o comportamento do sistema.

Para **identificar cada processo elementar**, as atividades a seguir devem ser realizadas:

- Compor e/ou decompor os Requisitos Funcionais do Usuário até a menor unidade de atividade que satisfaz todos os itens a seguir:
  - é significativo para o usuário
  - constitui uma transação completa
  - é auto-contido e
  - deixa o negócio de aplicação sendo medida em um estado consistente
- Identifique um processo elementar para cada unidade de atividade identificada que agrupa todos os critérios acima.

Para **classificar um processo elementar** como uma Entrada Externa (EE), Saída Externa (SE) ou uma Consulta Externa (CE), deve se basear na sua intenção primária:

- Classificar como uma **EE** se
  - tem a intenção primária de manter um ou mais ALIs ou alterar o comportamento da aplicação e
  - incluir a lógica de processamento para aceitar dados ou informações de controle que entra na fronteira da aplicação.
- Classificar como uma **SE** se
  - tem a intenção primária de apresentar informações ao usuário e
  - incluir pelo menos uma das seguintes formas de lógica de processamento: cálculos matemáticos são realizados; um ou mais ALIs são atualizados; é criado dado derivado ou; o comportamento da aplicação é alterado.
- Classificar como uma **CE** se
  - tem a intenção primária de apresentar informações ao usuário e
  - referenciar uma função de dados para recuperar dados ou informações de controle e não satisfaz o critério de ser classificado como uma SE.

Função	Tipo da Função de Transação		
	EE	SE	CE
Alterar o comportamento do sistema	IP	F	N/A
Manter um ou mais ALIs	IP	F	N/A
Apresentar a informação ao usuário	F	IP	IP

Legenda:

- **IP** a intenção primária do tipo de função de transação
- **F** é uma função do tipo de função de transação, mas não é a intenção primária e está presente algumas vezes
- **N/A** a função não é permitida para o tipo de função de transação

O número de EEs, SEs e CEs e suas complexidades funcionais determinam a contribuição das funções de transação para o tamanho funcional.

A **complexidade funcional** das funções de transações é medida com base na quantidade de tipos de arquivos referenciados (ALRs) e de tipos de dados elementares (DERs) associados a cada EE, SE e CE.

- Um **Tipo de Dado Elementar (DER)** é um campo único, reconhecido pelo usuário e não repetido.
- Um **Tipo de Arquivo Referenciado (ALR)** é uma função de dados lida e/ou mantida pela função de transação.

A complexidade funcional de cada função de transação deve ser determinada usando o número de RLRs e DERs de acordo com as matrizes a seguir:

Entrada Externas:

	1 a 4 DERs	5 a 15 DERs	16 ou mais DERs
0 a 1 RLRs	Baixa	Baixa	Média
2 RLRs	Baixa	Média	Alta
3 ou mais RLRs	Média	Alta	Alta

Saída Externas e Consultas Externas:

	1 a 5 DERs	6 a 19 DERs	20 ou mais DERs
0 a 1 RLRs	Baixa	Baixa	Média
2 a 3 RLRs	Baixa	Média	Alta
4 ou mais RLRs	Média	Alta	Alta

O tamanho funcional de cada função de transação é determinado usando o tipo e a complexidade funcional de acordo com a tabela a seguir:

Tamanho das funções de transação

		Tipo		
		EE	**SE**	**CE**
Complexidade Funcional	Baixa	3	4	3
	Média	4	5	4
	Alta	6	7	6

EXEMPLO

Para calcular o tamanho funcional das funções de transações, basta multiplicar o número de funções de cada tipo, pelos seus respectivos tamanhos funcionais. Vejamos um exemplo de uma aplicação que possua:

- 1 EE de complexidade alta.
- 2 EE de complexidade média.
- 5 EE de complexidade baixa.
- 2 SE de complexidade alta.
- 5 SE de complexidade média.
- 1 SE de complexidade baixa.
- 3 CE de complexidade alta.
- 4 CE de complexidade média.
- 2 CE de complexidade baixa.

Logo, para chegar no tamanho funcional:

- 1 EE de complexidade alta = 1 x 6 = 06
- 2 EE de complexidade média = 2 x 4 = 08
- 5 EE de complexidade baixa = 5 x 3 = 15
- 2 SE de complexidade alta = 2 x 7 = 14
- 5 SE de complexidade média = 5 x 5 = 25
- 1 SE de complexidade baixa = 1 x 4 = 04
- 3 CE de complexidade alta. = 3 x 6 = 18
- 4 CE de complexidade média. = 4 x 4 = 16
- 2 CE de complexidade baixa. = 2 x 3 = 06

TOTAL = 112

Então, o tamanho do software em questão, considerando apenas as funções de transações, é de 112 pontos de função.

3.2.5. Tamanho funcional

Considerando as funções de dados e as funções de transação, podemos usar a seguinte tabela para o cálculo do tamanho funcional:

	Baixa	Média	Alta
ALI	7	10	15
AIE	5	7	10
SE	4	5	7
CE	3	4	6
EE	3	4	6

3.2.6. Medir a funcionalidade de conversão

O escopo da contagem de um projeto de desenvolvimento ou melhoria também pode incluir o tamanho funcional da funcionalidade de conversão requerida para o mesmo.

3.2.7. Medir a funcionalidade correspondente a melhorias

O tamanho funcional da aplicação poderá ser atualizado para refletir:

1) a funcionalidade incluída, que aumentará o tamanho funcional da aplicação;
2) a funcionalidade alterada, que poderá aumentar, diminuir ou não ter efeito sobre o tamanho funcional da aplicação;
3) a funcionalidade excluída, que diminuirá o tamanho funcional da aplicação.

3.2.8. Calcular o tamanho funcional

O objetivo e escopo da contagem deverão ser considerados na seleção e utilização da fórmula apropriada para calcular o tamanho funcional.

- O tamanho funcional de um **projeto de desenvolvimento** deverá ser calculado utilizando-se a Fórmula:
    - **DFP = ADD + CFP**, onde
        - **DFP** é a contagem de pontos de função do projeto de desenvolvimento;
        - **ADD** é o tamanho das funções a serem entregues ao usuário pelo projeto de desenvolvimento;
        - **CFP** é o tamanho da funcionalidade de conversão.

- O tamanho funcional de uma **aplicação**, medido após o projeto de desenvolvimento, ou a qualquer tempo no ciclo de vida da aplicação deverá ser calculado utilizando-se a Fórmula:
    - **AFP = ADD**, onde
        - **AFP** é a contagem de pontos de função da aplicação;
        - **ADD** é o tamanho das funções a serem entregues ao usuário pelo projeto de desenvolvimento (excluído o tamanho de qualquer funcionalidade de conversão), ou a funcionalidade existente no momento da contagem da aplicação.
- O tamanho funcional de um **projeto de melhoria** deverá ser calculado utilizando-se a Fórmula:
    - **EFP = ADD + CHGA + CFP + DEL**, onde
        - **EFP** é a contagem de pontos de função do projeto de melhoria;
        - **ADD** é o tamanho das funções incluídas pelo projeto de melhoria;
        - **CHGA** é o tamanho das funções alteradas pelo projeto de melhoria – conforme as mesmas estão / estarão após a implementação;
        - **CFP** é o tamanho da funcionalidade de conversão;
        - **DEL** é o tamanho das funções excluídas pelo projeto de melhoria.
- O tamanho funcional de uma **aplicação após um projeto de melhoria** deverá ser calculado utilizando-se a Fórmula:
    - **AFPA = (AFPB + ADD + CHGA) - (CHGB + DEL)**, onde
        - **AFPA** é a contagem de pontos de função da aplicação após o projeto de melhoria;
        - **AFPB** é a contagem de pontos de função da aplicação antes do projeto de melhoria;
        - **ADD** é o tamanho das funções incluídas pelo projeto de melhoria;
        - **CHGA** é o tamanho das funções alteradas pelo projeto de melhoria – como estão / estarão após a implementação;
        - **CHGB** é o tamanho das funções alteradas pelo projeto de melhoria – como estão / estavam antes do início do projeto;
        - **DEL** é o tamanho das funções excluídas pelo projeto de melhoria.

3.2.9. Documentar a contagem dos pontos de função

A contagem de pontos de função deve ser documentada como segue:

- o propósito e o tipo da contagem;
- o escopo da contagem e a fronteira da aplicação;
- a data da contagem;
- uma lista de todas as funções de dados e de transação, incluindo o respectivo tipo e complexidade, bem como o número de pontos de função atribuído a cada uma;
- o resultado da contagem;
- quaisquer suposições feitas e questões resolvidas.

3.2.10. Reportar o resultado da contagem de pontos de função

Os resultados que mantenham conformidade com o Padrão IFPUG deverão ser reportados como segue: S FP (IFPUG–IS), onde S é S é o resultado da contagem de pontos de função; FP é a unidade de tamanho do método FSM do IFPUG; IS é este Padrão Internacional (ISO/IEC 20926:200x).

3.2.11. Calcular fator de ajuste

Algumas pessoas podem utilizar um **valor de fator de ajuste (VAF)**, que **considera as 14 características gerais do sistema (GSCs)**. Cada característica tem descrições associadas que ajudam a determinar o nível de influência da característica. O **nível de influência de cada característica varia em uma escala de 0 a 5**, de sem influência até forte influência.

As 14 características gerais do sistema estão resumidas no Fator de Ajuste. Quando aplicado, o **fator de ajuste ajusta o tamanho funcional não ajustado em +/- 35%** para produzir o tamanho funcional ajustado.

As 14 características gerais do sistema são:

Com base nos requisitos estabelecidos pelo usuário, cada característica geral do sistema (CGS) deve ser avaliada em termos de seus **níveis de influência (NI)** em uma **escala de 0 a 5**:

Pontue como	Influência no Sistema
0	Não presente ou sem influência
1	Influência Mínima
2	Influência Moderada
3	Influência Média
4	Influência Significativa
5	Forte influência

O valor de fator de ajuste é então calculado com base na seguinte fórmula **VAF = (TDI * 0,01) + 0,65**, onde TDI é o Total do Nível de Influência. Esse valor pode ser multiplicado pela quantidade de pontos de função não ajustados para chegar a tamanho do software em pontos de função ajustados.

EXEMPLO

Considere que ao calcular o tamanho funcional do software, havia se chegado a um total de 150 pontos de função. O fator de ajuste foi aplicado segundo as seguintes características e níveis respectivos:

1. Comunicação de Dados	5
2. Processamento Distribuído	4
3. Performance	4
4. Configuração Intensamente Utilizada	5
5. Volume de Transações	3
6. Entrada de Dados On-Line	2
7. Eficiência do Usuário Final	2
8. Atualização On-Line	1
9. Processamento Complexo	1
10. Reusabilidade	1
11. Facilidade de Instalação	5
12. Facilidade de Operação	5
13. Múltiplos Locais	4
14. Facilidade de Mudança	3
	TOTAL 45

Logo, para chegar no tamanho funcional ajustado:

VAF = (TDI * 0,01) + 0,65 = (45 *0,01) + 0,65 = 0,45 + 0,65 = 1,10

Logo, total de pontos de função ajustados é 150 x 1,10 = 165

REFERÊNCIAS

IFPUG. **Manual de Práticas de Contagem de Pontos de Função**. Versão 4.3.1.

4. UNIDADE 4 – CMMI

4.1. Visão Geral do CMMI

O **CMMI (Modelo Integrado de Maturidade e Capacidade)** é um conjunto integrado de melhores práticas que permite às empresas melhorar o desempenho de seus principais processos de negócios. Este modelo foi desenvolvido por equipes de produtos com membros da indústria e do CMMI Institute.

Na sua essência, o CMMI fornece um roteiro claro para construir, melhorar e sustentar a capacidade. Dentre as principais razões para desenvolver a capacidade, temos:

- Maior satisfação do cliente;
- Maior probabilidade de captar novos negócios e repetir negócios;
- Maior lucro através de melhor qualidade e menos retrabalho.
- Maior produtividade.
- Diminuição de riscos.

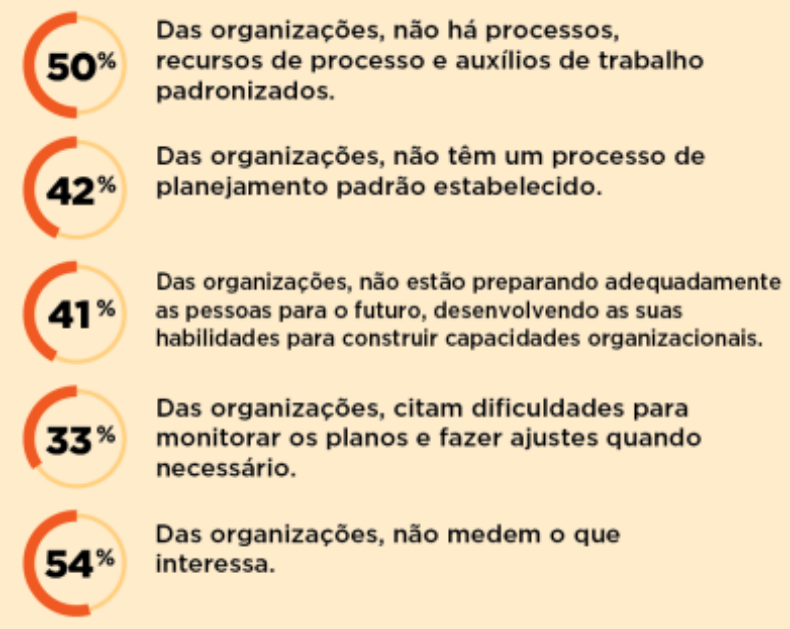
A arquitetura e o desenho do Conjunto de Produtos do CMMI 2.0 são um distanciamento radical de seus antecessores para torná-lo mais útil e adotável por clientes e empresas. A arquitetura do CMMI foi projetada especificamente para ser flexível, ágil e evoluir à medida que esses e outros fatores mudam. Isso permite o rápido desenvolvimento e a adição de novos conteúdos relevantes na velocidade dos negócios, tecnologia e alterações.

O CMMI fornece orientação para aplicar esse conjunto de melhores práticas em uma empresa ou organização, para garantir soluções de qualidade e oportunas que encantam os clientes e usuários finais. Qualquer empresa ou organização pode se beneficiar da melhoria de desempenho e da redução de riscos. O CMMI fornece um roteiro que orienta a melhoria de atividades ad hoc e de processos disciplinados e consistentes para realização de objetivos de negócios relacionados a desempenho, qualidade, custo, cronograma e funcionalidade.

4.1.1. Por que usar o CMMI?

O **CMMI** ajuda a empresa a entender o seu atual nível de capacidade e desempenho. Se as necessidades e os objetivos de negócios não estiverem sendo atendidos, as práticas do CMMI podem pautar a melhoria para elevar e otimizar o desempenho. Concentrando-se principalmente nos benefícios de negócios e na melhoria dos processos impulsionadores de desempenho para melhor atender às necessidades da empresa e, em última instância, do cliente.

Uma pesquisa concluiu que quase metade das organizações pesquisadas não possui processos padrão que lhes permitam adaptar-se rapidamente. Dentre os principais resultados dessa pesquisa, destacam-se os seguintes:



4.1.2. Benefícios do CMMI

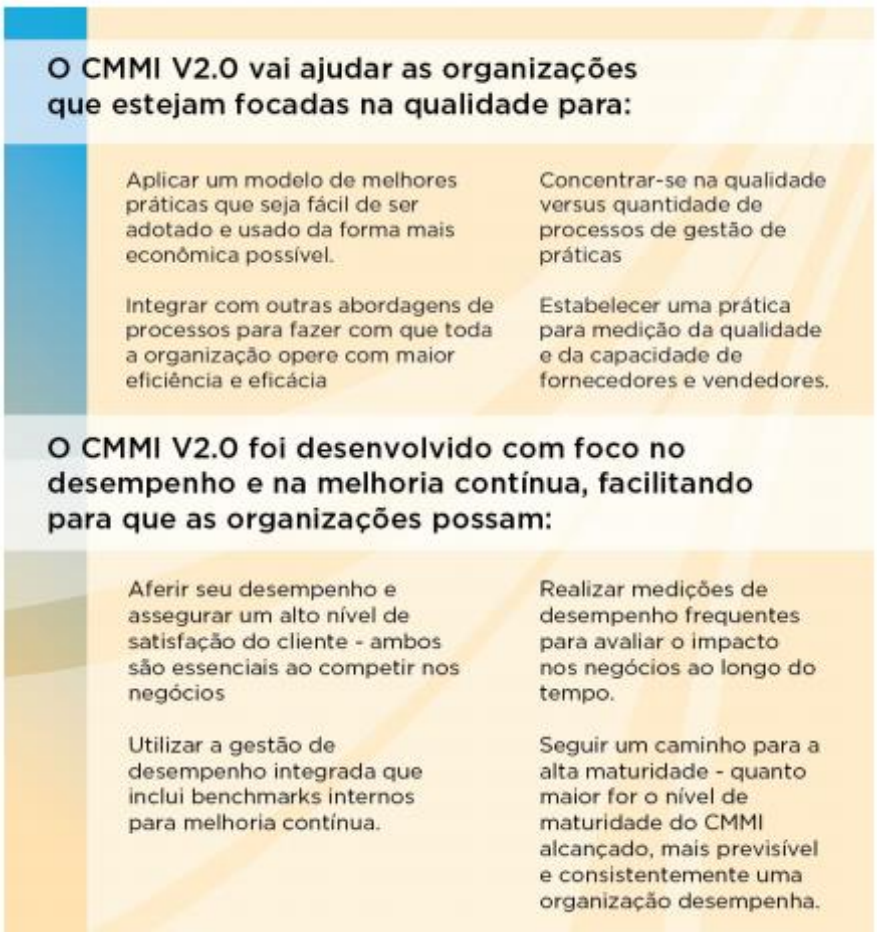
O uso do CMMI oferece muitos benefícios, incluindo:

- Um retorno positivo dos investimentos em melhoria de processos e desempenho.
- Cumprimento de compromissos que resultam em:
  - Entrega mais pontual
  - Menos crises de última hora
  - Melhor controle de custos
  - Maior qualidade das soluções
- Visibilidade de gerenciamento que resulta em:
  - Resposta mais rápida a problemas e riscos
  - Menos surpresas
  - Soluções de alta qualidade que atendem às necessidades e expectativas dos clientes
  - Redução nas reclamações de clientes
  - Redução de retrabalho
  - Menor rotatividade de colaboradores

4.1.3. Melhorar o desempenho

É importante entender o atual nível de desempenho da organização e a extensão em que ela se alinha às atuais necessidades e objetivos da empresa. Se o desempenho não estiver atendendo às necessidades e objetivos da empresa, então a **melhoria do processo é usada para elevar o desempenho até o nível necessário**. O desempenho deve ser gerenciado em todos os níveis do negócio e deve ser um dos principais impulsionadores de mudança do processo.

A figura a seguir ilustra como o CMMI aborda a melhoria do desempenho e da capacidade:



4.1.4. Objetivo do CMMI

O modelo **CMMI** é um conjunto organizado de melhores práticas para a melhoria e desempenho dos negócios. As melhores práticas no modelo se concentram no que precisa ser feito para melhorar o desempenho, não como fazê-lo. A adoção correta do CMMI é dependente de cada situação. Uma abordagem específica pode não funcionar em todas as situações. O CMMI foi explicitamente projetado para ser compreensível, acessível e flexível para uma ampla variedade de empresas e tipos de trabalho.

4.1.5. Público do CMMI

O **público do CMMI** inclui qualquer pessoa interessada em melhorar o desempenho em qualquer ambiente de negócios. Quer você esteja procurando informações para começar a melhorar o seu desempenho ou já esteja familiarizado com o conceito de modelos de maturidade e capacidade, o CMMI pode ser útil para você. O CMMI também pode ser usado com eficiência para realizar auditorias na seleção de fornecedores potenciais ou em uma organização que você possa ter interesse em adquirir.

4.1.6. Estrutura e conteúdo do modelo

Em vez de ter todo o material contido em um único documento linear ou livro, o modelo CMMI dá suporte no fornecimento de conteúdo a usuários em diferentes formatos, diferentes níveis de detalhe e com diferente ênfase dependendo das visualizações definidas ou necessárias. Isso permite um conjunto robusto de informações adaptadas para uso e necessidades específicas.

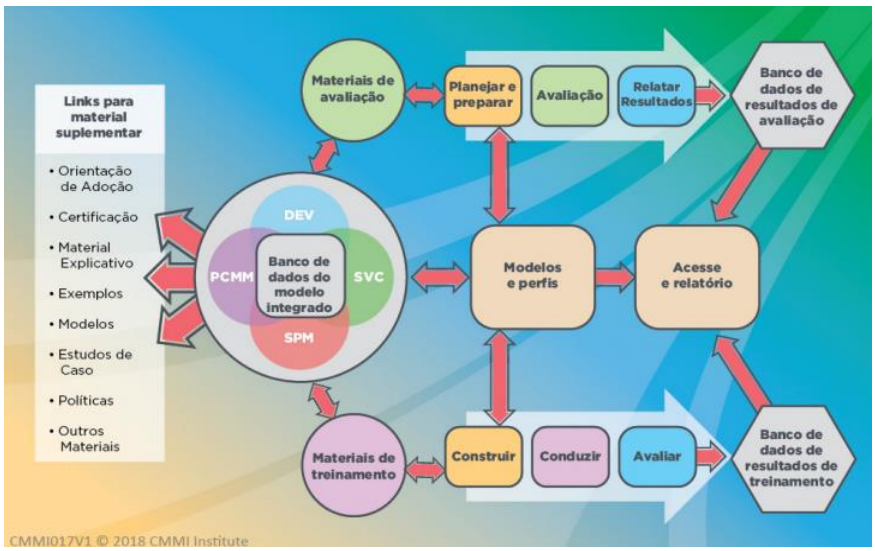
Para facilitar a apresentação do conteúdo do modelo de uma maneira mais acessível e fácil de entender e facilitar atualizações futuras, o conteúdo do modelo foi dividido em **6 partes**, organizadas em três seções principais, incluindo a visão geral (esta seção), áreas de prática e Anexos.

Parte Número:	Título	Descrição
Seção: Visão Geral		
Parte 1	Sobre o CMMI V2.0	Fornece uma visão geral do Conjunto de Produtos CMMI V2.0, incluindo um Sumário Executivo do modelo.
Parte 2	Adoção correta do CMMI	Fornece o contexto para o entendimento e uso do CMMI de uma forma que atinja os resultados de negócios tangíveis.
Parte 3	Persistência e hábito do processo	Descreve como criar e manter a melhoria e a capacidade de desempenho dos negócios em toda a organização.
Parte 4	Alcançando a alta maturidade	Baseia-se no sucesso inicial do desempenho ao adotar o modelo e o eleva para otimizar o desempenho através da compreensão dos objetivos de variação e desempenho.
Seção: Áreas de prática		
Parte 5	Áreas de prática	Contendo a maior parte do conteúdo do modelo, esta parte contém todas as áreas de prática do modelo.
Seção: Anexo		
Parte 6	Anexos A-I	Informações mais detalhadas sobre a adoção do CMMI, compreensão das visualizações e níveis predefinidos que ele contém e como funcionam, além do glossário, lista de acrônimos, índice e reconhecimentos.

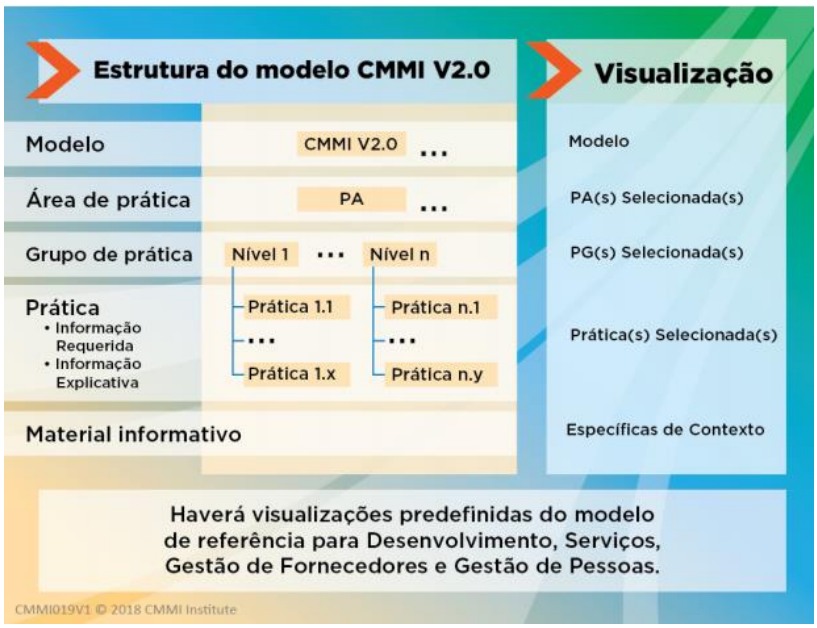
4.1.7. Conjunto de produtos CMMI V2.0

Em sua essência, o CMMI é um conjunto de visualizações predefinidas e personalizadas que se aplicam a diferentes ambientes de negócios. O Conjunto de Produtos CMMI V2.0 contém **cinco componentes**, conforme figura a seguir:

A figura a seguir mostra todo o conjunto integrado de produtos CMMI V2.0. Essa abordagem integrada foi projetada para reduzir a "independência" dos componentes do conjunto de produtos.



Com exceção da visão geral do modelo e dos anexos no interior do modelo, o formato do material segue uma estrutura modular comum. A Figura a seguir mostra a visão geral de alto nível da estrutura modular do modelo CMMI.



Vamos descrever os principais componentes:

- **Visualização**: seleção de um conjunto de componentes do modelo importantes e selecionados pelo usuário final ou predefinidos pelo CMMI Institute, tais como:
  - CMMI Desenvolvimento
  - CMMI Serviços
  - CMMI Gestão de Fornecedores
  - Área de Planejamento e Gestão de Capacidade de Trabalho do CMMI
- **Área de Prática**: uma coleção de práticas similares que, conjuntamente, alcançam a intenção definida, valor e as informações requeridas descritas nessa Área de Prática.
- **Grupo de Prática**: estrutura organizadora (ex., nível evolutivo) para as práticas dentro de uma Área de Prática para ajudar na compreensão e adoção e fornecer um caminho para melhoria do desempenho.
- **Prática**: consiste de duas partes:
  - **Informações requeridas**: para entender a plena intenção e o valor da prática, que inclui a declaração de prática, a declaração de valor e todas as informações adicionais requeridas entre ela e as informações explicativas da prática.

- **Informações Explicativas**: partes remanescentes da prática, incluindo exemplos de atividades e produtos de trabalho, que são importantes e úteis para melhor entender o significado e a intenção das informações requeridas. Não podem ser ignoradas, pois são necessárias para entender corretamente a prática.

- **Material informativo**: inclui tudo o mais no modelo, exceto as informações requeridas nas práticas. As informações explicativas fazem parte do material informativo. O material informativo inclui a visão geral e os apêndices (ex., glossário, índice, etc.). Os links externos podem ser adicionados ao material informativo. Estes links podem ser para material informativo adicional, orientação de adoção, exemplo de adoção, orientação de transição de um modelo ou padrão para outros, modelos e materiais de treinamento.

4.1.8. Área de capacidade

Uma **área de capacidade** é um grupo de áreas de prática relacionadas que podem proporcionar um melhor desempenho nas habilidades e atividades de uma organização ou projeto. Uma visualização da área de capacidade é um subconjunto do modelo CMMI que descreve um conjunto predefinido de áreas de prática que compõem uma área de capacidade específica. As áreas de capacidade são um tipo de visualização.

4.1.9. Categorias para as áreas de capacidade

As **categorias** são grupos lógicos ou visualizações de áreas de capacidade relacionadas que abordam problemas comuns encontrados pelas empresas ao produzir ou fornecer soluções. Uma das lições aprendidas com a experiência da indústria é que a criação de pequenos grupos de tópicos semelhantes em uma lista os torna mais fáceis para entender e lembrar.

As categorias são:

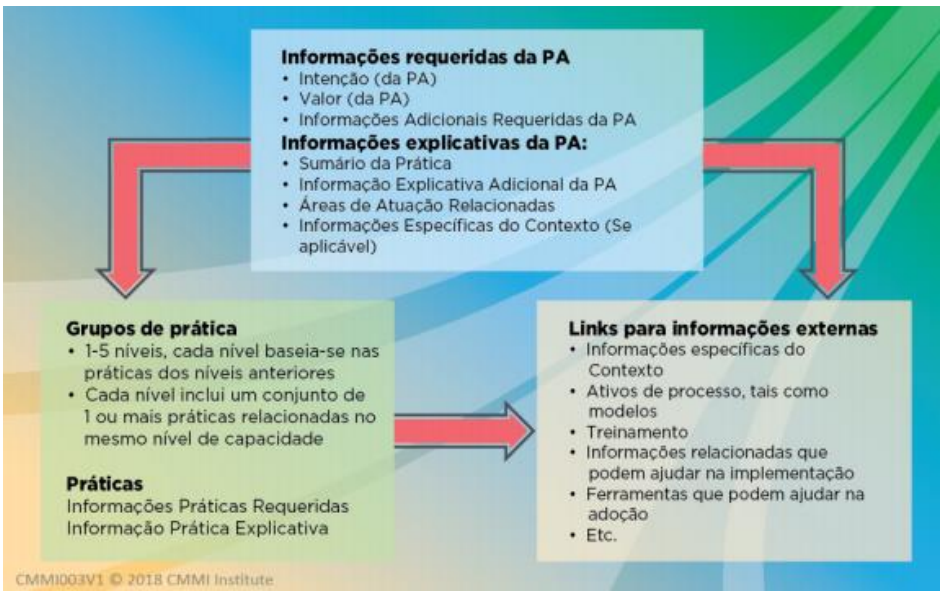
- **Execução** - Áreas de capacidade para produção e entrega de soluções de qualidade
- **Gestão** - Áreas de capacidade para planejar e gerenciar a implementação de soluções
- **Habilitação** - Áreas de capacidade para prestar suporte à implementação e entrega de soluções
- **Melhoria** - Áreas de capacidade para sustentar e melhorar o desempenho

Vejamos as categorias e áreas de capacidade associadas:

4.1.10. Área de prática

Uma **área de prática** é um conjunto de práticas que descrevem coletivamente as atividades críticas necessárias para alcançar uma intenção e um valor definidos.

As áreas de prática do CMMI estão organizadas conforme a figura a seguir:

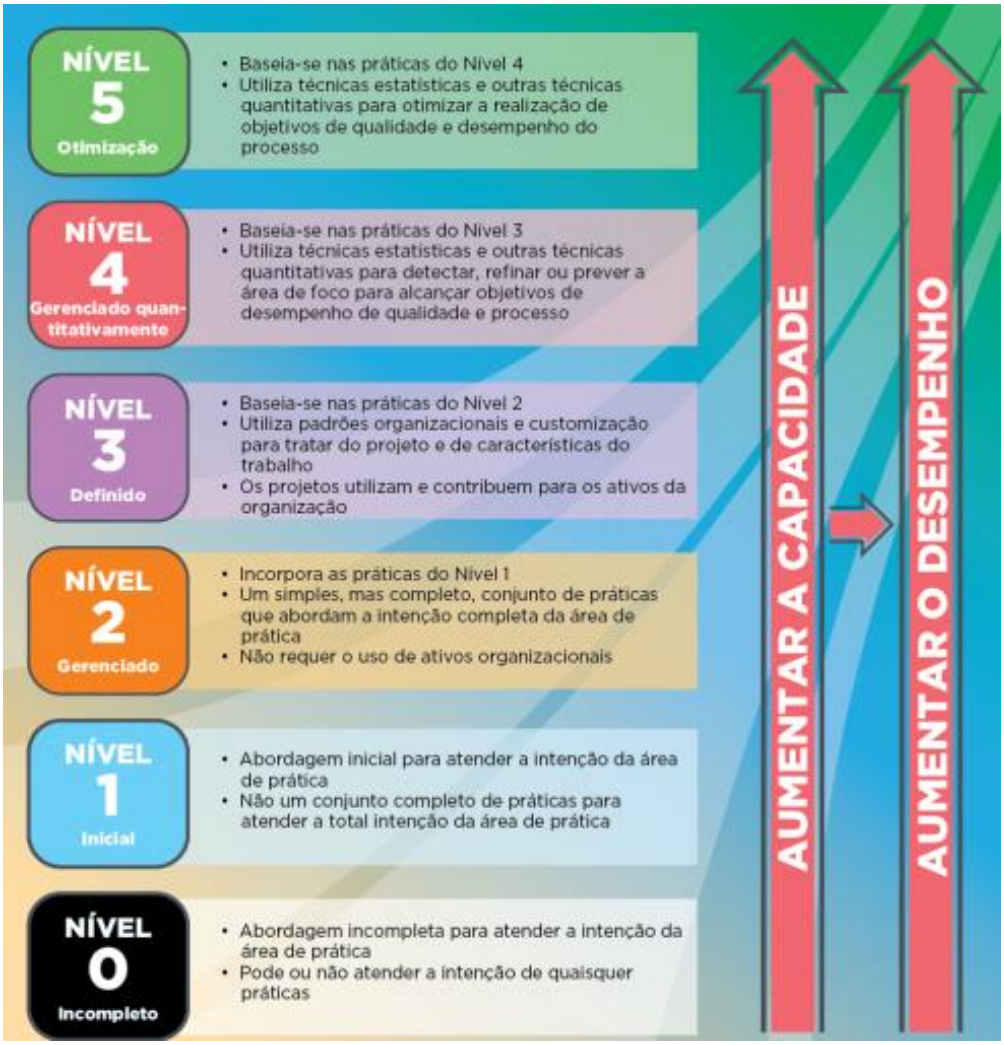


4.1.11. Níveis evolutivos

Dentro das áreas de prática, as práticas são organizadas em um conjunto de **níveis evolutivos** chamados Nível 1, Nível 2 etc., que fornecem um caminho para a melhoria do desempenho.

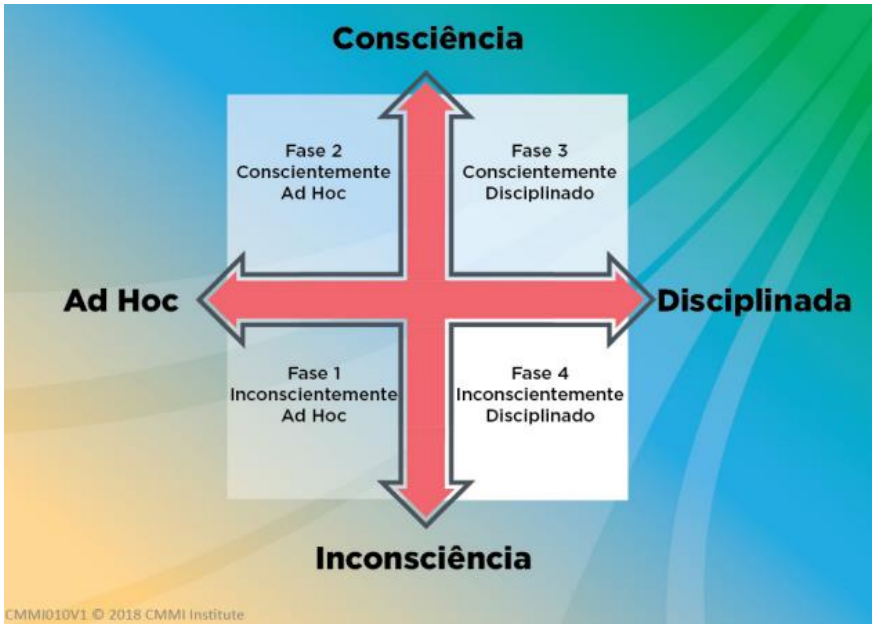
Cada nível evolutivo baseia-se nos níveis anteriores, adicionando um novo recurso ou sofisticação que aumentam a capacidade.

Os níveis evolutivos são resumidos na figura a seguir:



4.1.12. Estágios da disciplina de processos

A base da melhoria de processos é incutir disciplina na cultura da organização. Isso inclui a forma como o trabalho é percebido, concluído e melhorado. A Figura a seguir mostra os quatro estágios da disciplina de processos pelos quais as organizações geralmente passam ao implementar a melhoria de processos. À medida que a organização avança pelos estágios, a capacidade e maturidade do processo aumentam, conduzindo a um melhor desempenho.



No **primeiro estágio**, a execução de processos é ad hoc e indisciplinada. As pessoas seguem seus próprios processos, não registrados, o que resulta em resultados variados e impedem o aprendizado organizacional sistêmico. A necessidade de melhoria de desempenho é reconhecida, mas a capacidade de melhorar é limitada e só é alcançada involuntariamente.

No **segundo estágio**, há uma compreensão consciente de que os processos são ad hoc e não registrados, limitando tanto a consistência de execução como a capacidade da organização de melhorar seu desempenho e seus processos.

No **terceiro estágio**, registre os processos e estabeleça mecanismos para garantir a fidelidade de execução do processo. As estruturas de suporte organizacional, incluindo a supervisão consistente da alta direção, incentivam o uso continuado dos processos e melhorias associadas.

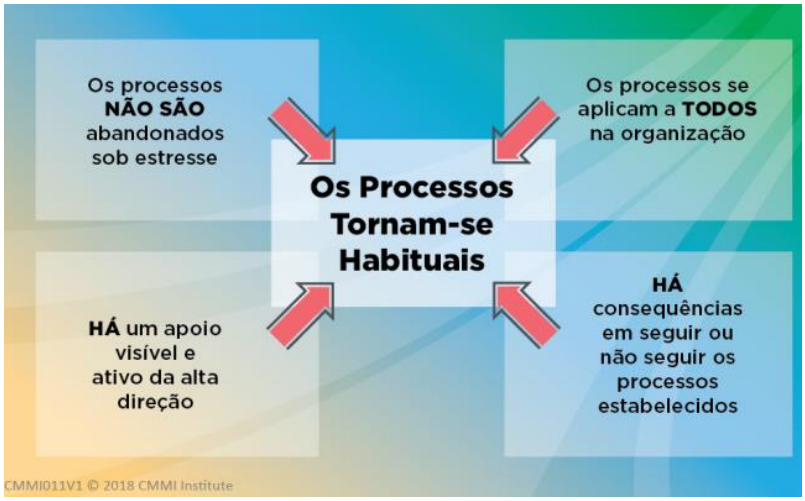
O **quarto e último estágio** é alcançado quando os processos e o desempenho estiverem claramente compreendidos, seguidos, persistentes e habituais.

4.1.13. Persistência e hábito do processo

No modelo CMMI, o termo “persistente e habitual” descreve o modo rotineiro de se fazer negócios e acompanhar e melhorar os processos que uma organização usa como parte da sua cultura corporativa:

- **Persistência**: Continuação estável ou teimosa em uma ação apesar da dificuldade ou oposição.
- **Hábito**: Uma tendência ou prática, especialmente aquela que é difícil de ser deixada.

A Figura a seguir descreve as quatro principais características para entender e criar práticas persistentes e habituais dentro de uma organização.

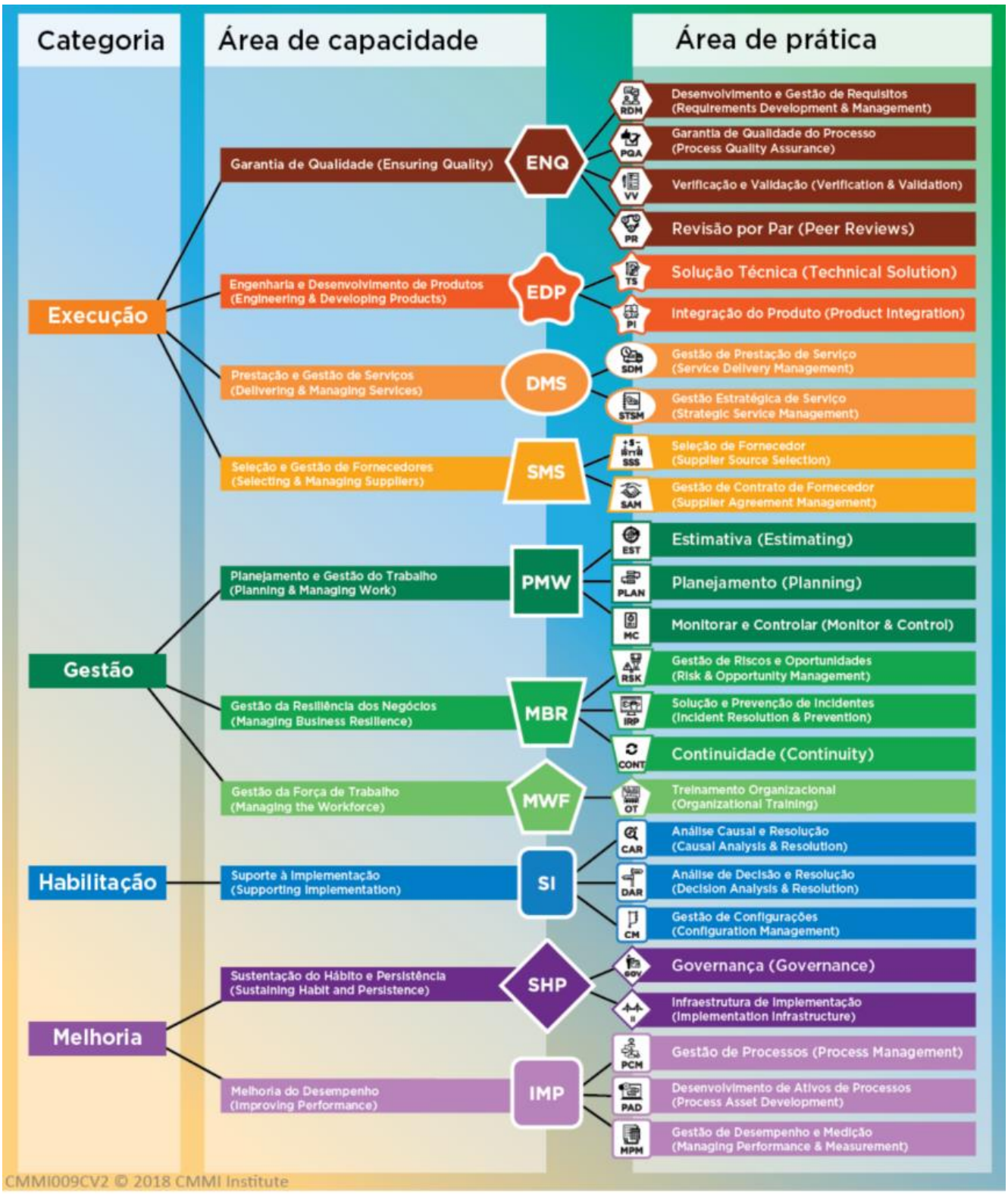


4.1.14. Alcançando a alta maturidade

No conjunto de produtos CMMI, o termo “alta maturidade” implica o uso de técnicas estatísticas e outras técnicas quantitativas em processos selecionados. A alta maturidade representa uma mudança fundamental na compreensão, gestão e melhoria de processos. À medida que as organizações avançam na maturidade de processos, elas adquirem um conhecimento mais profundo de como os processos são usados e interagem, o que lhes dá uma clara vantagem competitiva. Organizações de alta maturidade demonstram um compromisso mais profundo com a melhoria de capacidades com foco na melhoria contínua do desempenho.

4.2. Áreas de Prática

As áreas de práticas do CMMI são organizadas por categoria e área de capacidade, conforme a figura a seguir:



4.2.1. Categoria de Execução

Na Categoria de Execução, temos as seguintes áreas de prática:

Área de Capacidade: Garantia da Qualidade		
Área de Prática	**Intenção**	**Valor**
Desenvolvimento e Gestão de Requisitos (Requirements Development and Management, RDM)	Esclarecer os requisitos, assegurar o entendimento comum pelas partes interessadas e alinhar os requisitos, planos e produtos de trabalho.	Garantir que as necessidades e expectativas dos clientes sejam satisfeitas
Garantia de Qualidade do Processo (Process Quality Assurance, PQA)	Verificar e facilitar a melhoria da qualidade dos processos realizados e dos produtos de trabalho resultantes.	Aumenta o uso consistente e a melhoria dos processos para maximizar o benefício de negócios e a satisfação do cliente.
Verificação e Validação (Verification and Validation, VV)	A verificação e validação inclui atividades que: • Confirmam se as soluções e componentes selecionados atendem aos requisitos • Demonstram soluções e componentes selecionados para atender à utilização pretendida em seu ambiente de destino	A verificação e validação de soluções e componentes selecionados em todo o projeto aumenta a probabilidade de a solução satisfazer o cliente.
Revisão por Par (Peer Reviews, PR)	Identificar e resolver problemas dos produtos de trabalho através de revisões por pares do produtor ou de Especialistas no Assunto (SMEs).	Reduzir custos e retrabalho detectando os problemas ou defeitos antecipadamente.

Área de Capacidade: Engenharia e Desenvolvimento de Produtos		
Área de Prática	**Intenção**	**Valor**
Solução técnica (Technical Solution, TS)	Projetar e criar soluções que atendam aos requisitos do cliente.	Proporciona um desenho de projeto e uma solução econômica que atenda aos requisitos do cliente e reduzem o retrabalho.
Integração do Produto (Product Integration, PI)	Integrar e fornecer a solução que atenda aos requisitos de funcionalidade e qualidade.	Aumenta a satisfação dos clientes, proporcionando solução que cumpra ou exceda seus requisitos de funcionalidade e qualidade.

Área de Capacidade: Prestação e Gestão de Serviços		
Área de Prática	**Intenção**	**Valor**
Gestão de Prestação de Serviço (Service Delivery Management, SDM)	Executar serviços e gerenciar o sistema de prestação de serviços	Aumenta a satisfação do cliente, prestando serviços que atendam ou superem suas expectativas.
Gestão Estratégica de Serviço (Strategic Service Management, STSM)	Desenvolver e implantar serviços padrão que sejam compatíveis com as necessidades e planos estratégicos de negócios.	Aumenta a probabilidade de atingir os objetivos de negócios, alinhando os serviços padrão com as necessidades do cliente.

Área de Capacidade: Seleção e Gestão de Fornecedores		
Área de Prática	**Intenção**	**Valor**
Seleção de Fornecedor (Supplier Source Selection, SSS)	Desenvolver e manter atualizado um pacote de materiais utilizado para buscar propostas de fornecedores potenciais e selecionar um ou mais fornecedores para fornecer a solução.	Melhora a capacidade de selecionar os fornecedores mais qualificados para fornecer soluções.
Gestão de Contrato de Fornecedor (Supplier Agreement Management, SAM)	Chegar a um acordo com os fornecedores selecionados, assegurar que o fornecedor e o adquirente atuem de acordo com os termos no decorrer do contrato, e avaliar os entregáveis do fornecedor.	Oferece um entendimento explícito entre o adquirente e o fornecedor para maximizar o sucesso dos esforços acordados para o fornecimento de um entregável do fornecedor.

4.2.2. Categoria de Gestão

Na Categoria de Gestão, temos as seguintes áreas de prática:

Área de Capacidade: Planejamento e Gestão do Trabalho		
Área de Prática	**Intenção**	**Valor**
Estimativa (Estimating, EST)	Estimar o tamanho, o esforço, a duração e o custo do trabalho e os recursos necessários para desenvolver, adquirir ou entregar a solução.	A estimativa oferece uma base para assumir compromissos, planejar e reduzir a incerteza, o que permite ações corretivas antecipadas e aumenta a probabilidade de cumprir os objetivos.
Planejamento (Planning, PLAN)	Elaborar planos para descrever o que é necessário para realizar o trabalho dentro dos padrões e restrições da organização, como: • Orçamento • Cronograma • Demanda, capacidade e disponibilidade de recursos • Qualidade • Requisitos de funcionalidade • Riscos e Oportunidades	Otimiza o custo, a funcionalidade e a qualidade para aumentar a probabilidade de cumprir os objetivos.
Monitorar e Controlar (Monitor and Control, MC)	Explicar o progresso do projeto para que ações corretivas apropriadas possam ser tomadas quando o desempenho se desviar significativamente dos planos.	Aumenta a probabilidade de cumprir os objetivos, adotando ações antecipadas para ajustar desvios significativos de desempenho.

Área de Capacidade: Gestão da Resiliência dos Negócios		
Área de Prática	**Intenção**	**Valor**
Gestão de Riscos e Oportunidades (Risk and Opportunity Management, RSK)	Identificar, registrar, analisar e gerenciar os potenciais riscos ou oportunidades.	Mitiga os impactos adversos ou capitaliza os impactos positivos para aumentar a probabilidade de cumprir os objetivos.
Resolução e Prevenção de Incidentes (Incident Resolution and Prevention, IRP)	Resolver e evitar interrupções prontamente para manter os níveis de prestação de serviços.	Minimizar o impacto das interrupções para atender aos objetivos e compromissos com o cliente de maneira mais eficaz.
Continuidade (Continuity, CONT)	Planejar as atividades de mitigação para interrupções significativas nas operações de negócios para que o trabalho possa continuar ou ser retomado.	Permite a operação continuada quando da ocorrência de interrupções graves ou eventos catastróficos.

Área de Capacidade: Gestão da Força de Trabalho		
Área de Prática	**Intenção**	**Valor**
Treinamento Organizacional (Organizational Training, OT)	Desenvolver as habilidades e o conhecimento dos profissionais para que desempenhem suas funções com eficiência e eficácia.	Reforça as habilidades e o conhecimento dos indivíduos para melhorar o desempenho do trabalho organizacional.

4.2.3. Categoria de Habilitação

Na Categoria de Habilitação, temos as seguintes áreas de prática:

Área de Capacidade: Suporte à implementação		
Área de Prática	**Intenção**	**Valor**
Análise Causal e Resolução (Causal Analysis and Resolution, CAR)	Identificar as causas dos resultados selecionados e tomar medidas para evitar a recorrência de resultados indesejáveis ou assegurar a recorrência de resultados positivos.	Abordar problemas de causa raiz elimina o retrabalho e melhora diretamente a qualidade e a produtividade.
Análise de Decisão e Resolução (Decision Analysis and Resolution, DAR)	Fazer e registrar as decisões utilizando um processo registrado que analisa as alternativas.	Aumenta a objetividade da tomada de decisão e a probabilidade de selecionar a solução ideal.
Gestão de Configuração (Configuration Management, CM)	Gerenciar a integridade de produtos de trabalho utilizando identificação de configuração, controle de versão, controle de mudanças e auditorias.	Reduz a perda de trabalho e aumenta a capacidade de entregar a versão correta da solução para o cliente.

4.2.4. Categoria de Melhoria

Na Categoria de Melhoria, temos as seguintes áreas de prática:

Área de Capacidade: Sustentação do Hábito e Persistência		
Área de Prática	**Intenção**	**Valor**
Governança (Governance, GOV)	Fornece orientação para a alta direção sobre seu papel no patrocínio e governança de atividades do processo.	Minimiza o custo de implementação do processo, aumenta a probabilidade de alcançar os objetivos e assegura que os processos implementados deem suporte e contribuam para o sucesso do negócio.
Infraestrutura de Implementação (Implementation Infrastructure, II)	Garantir que os processos importantes para uma organização sejam contínua e habitualmente utilizados e melhorados.	Sustenta a capacidade de alcançar metas e objetivos consistentemente de forma eficiente e eficaz.

Área de Capacidade: Melhoria do Desempenho		
Área de Prática	**Intenção**	**Valor**
Gestão de Processos (Process Management, PCM)	Gerencia e implementa a melhoria contínua de processos e infraestrutura para: • Dar suporte à realização de objetivos de negócios • Identificar e implementar as melhorias de processo mais benéficas • Tornar visíveis os resultados da melhoria de processos, acessíveis e sustentáveis	Assegura que os processos, infraestrutura e sua melhoria contribuam para o sucesso dos objetivos de negócios.
Desenvolvimento de Ativos de Processos (Process Asset Development, PAD)	Desenvolver e manter em dia os ativos de processo necessários para realizar o trabalho.	Possibilita entender e repetir um desempenho bem-sucedido.
Gerenciamento de Desempenho e Medição (Managing Performance and Measurement, MPM)	Gerenciar o desempenho utilizando as medições e análises para alcançar os objetivos de negócios.	Maximiza o retorno dos negócios sobre o investimento, concentrando os esforços de gestão e melhoria no custo, cronograma e desempenho de qualidade.

REFERÊNCIAS

CMMI Institute. **Modelo CMMI 2.0**. 2018.

5. UNIDADE 5 – MPSBR

5.1. Introdução ao MPSBR

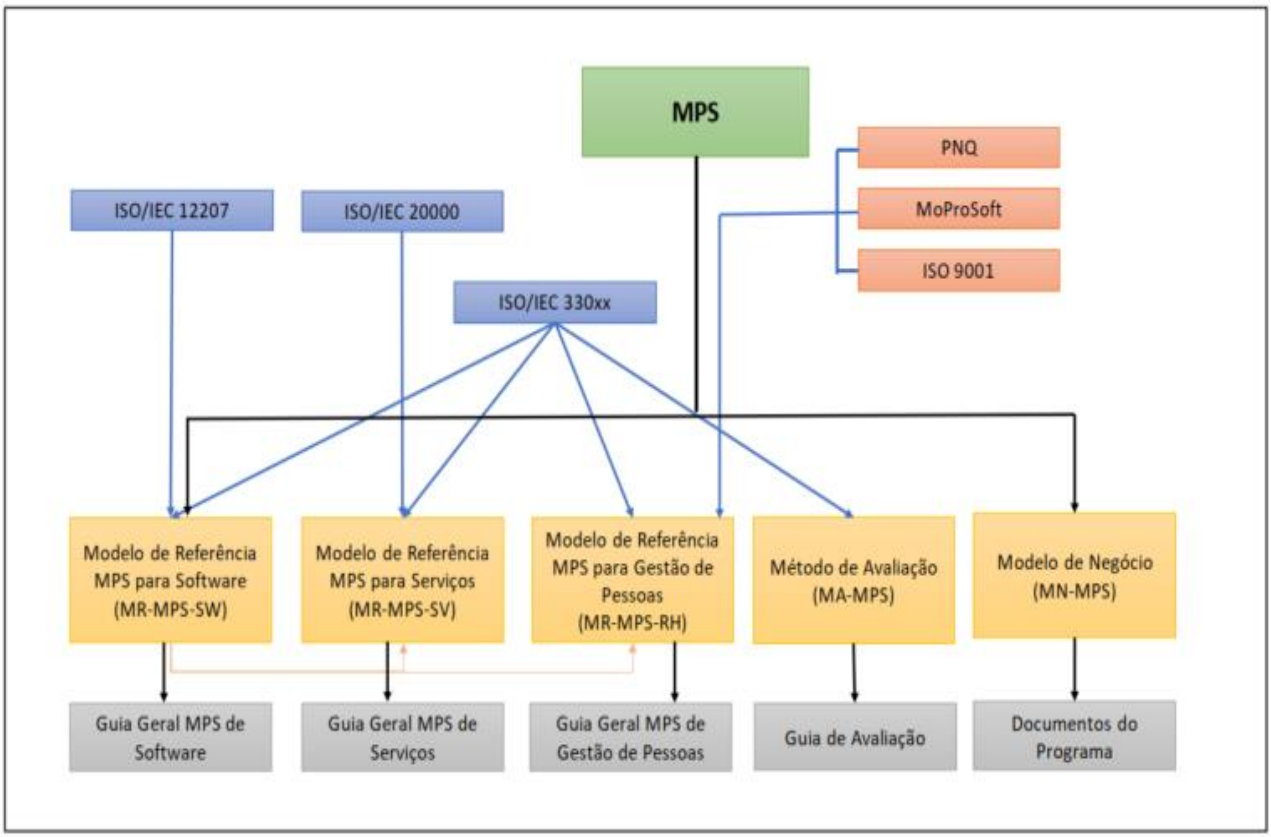
O **Programa MPS.BR** é um programa mobilizador, de longo prazo, criado em dezembro de 2003, coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), que teve o apoio do Ministério da Ciência, Tecnologia, Inovações e Comunicações (MCTIC), Financiadora de Estudos e Projetos (FINEP), Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE) e Banco Interamericano de Desenvolvimento (BID/FUMIN).

O **objetivo do programa MPS.BR** é o aumento da competitividade das organizações pela melhoria de seus processos. O programa tem duas metas a serem alcançadas a médio e longo prazos:

- **a) meta técnica**, visando ao aprimoramento do programa, com:
  - (i) atualização dos modelos e guias dos Modelos de Maturidade do MPS;
  - (ii) formação de consultores, instrutores de cursos, avaliadores e Instituições Implementadoras/Avaliadoras nos modelos de Maturidade do MPS.
- **b) meta de negócio**, visando à disseminação e viabilização na adoção dos Modelos de Maturidade do MPS para a melhoria da competitividade das empresas com:
  - (i) criação e aprimoramento do modelo de negócio MNMPS;
  - (ii) realização de cursos, provas e workshops MPS;
  - (iii) apoio para organizações que implementaram os Modelos MPS;
  - (iv) transparência para as organizações que realizaram a avaliação MPS.

5.2. Descrição geral do modelo MPS

O Programa MPS.BR possui cinco (5) componentes: Modelo de Referência MPS para Software (MR-MPS-SW), Modelo de Referência MPS para Serviços (MR-MPS-SV), Modelo de Referência MPS para Gestão de Recursos Humanos (MR-MPS-RH), Método de Avaliação (MA-MPS) e Modelo de Negócio (MN-MPS). Cada componente é descrito por meio de guias e/ou documentos do Programa MPS.BR.



Os componentes possuem algumas normas e modelos como base técnicas, entre as quais destacam-se a ISO/IEC 12207:2017, a ISO/IEC 20000, a família 330xx da ISO/IEC, a NBR ISO 9001:2015, o Prêmio Nacional de Qualidade (PNQ) e o MoProSoft.

Os modelos MPS estão descritos por meio de documentos em formato de guias, disponíveis em www.softex.br:

- **Guia Geral MPS de Software**: contém a descrição da estrutura dos modelos MPS e detalha o Modelo de Referência MPS para Software (MR-MPS-SW), seus componentes e as definições comuns necessárias para seu entendimento e aplicação.
- **Guia Geral MPS de Serviços**: contém a descrição da estrutura dos modelos MPS e detalha o Modelo de Referência MPS para Serviços (MR-MPS-SV), seus componentes e as definições comuns necessárias para seu entendimento e aplicação;
- **Guia Geral MPS de Gestão de Pessoas**: contém a descrição da estrutura dos modelos MPS e detalha o Modelo de Referência MPS para Gestão de Pessoas (MR-MPS-RH), seus componentes e as definições comuns necessárias para seu entendimento e aplicação;
- **Guia de Avaliação**: descreve o processo e o método de avaliação MA-MPS, os requisitos para avaliadores líderes, avaliadores adjuntos e Instituições Avaliadoras (IA).

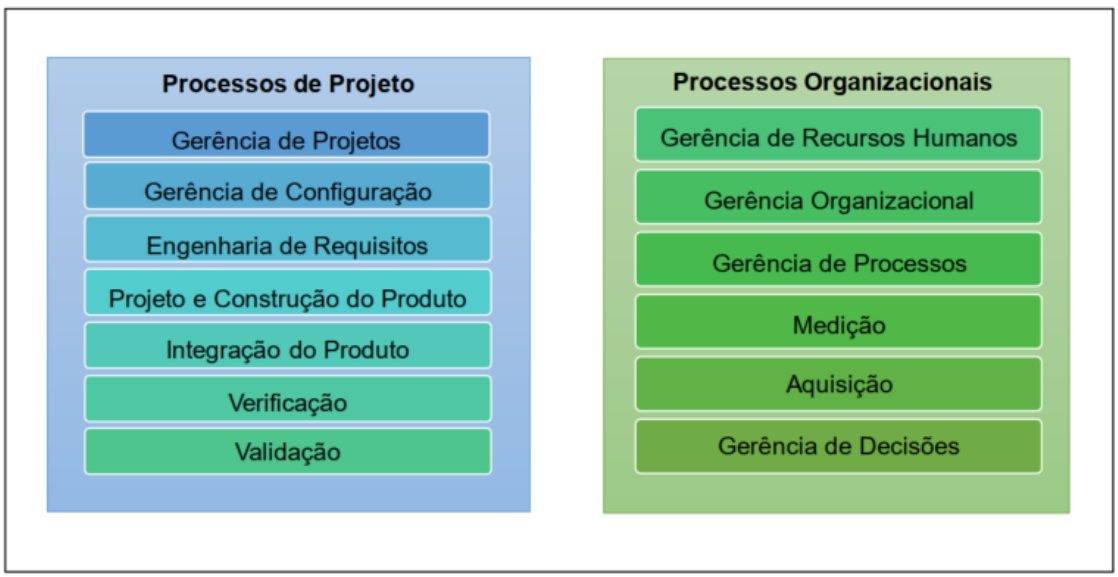
- **Modelo de Negócio MN-MPS**: descreve regras de negócio para:
  - (i) implementação dos modelos MPS pelas Instituições Implementadoras (II),
  - (ii) avaliação seguindo o MA-MPS pelas Instituições Avaliadoras (IA),
  - (iii) organização de grupos de empresas, para implementação e avaliação de acordo com os modelos MPS, pelas Instituições Organizadoras de Grupos de Empresas (IOGE) e
  - (iv) realização de treinamentos oficiais do MPS por meio de cursos, provas e workshops.

5.3. Descrição do MR-MPS-SW

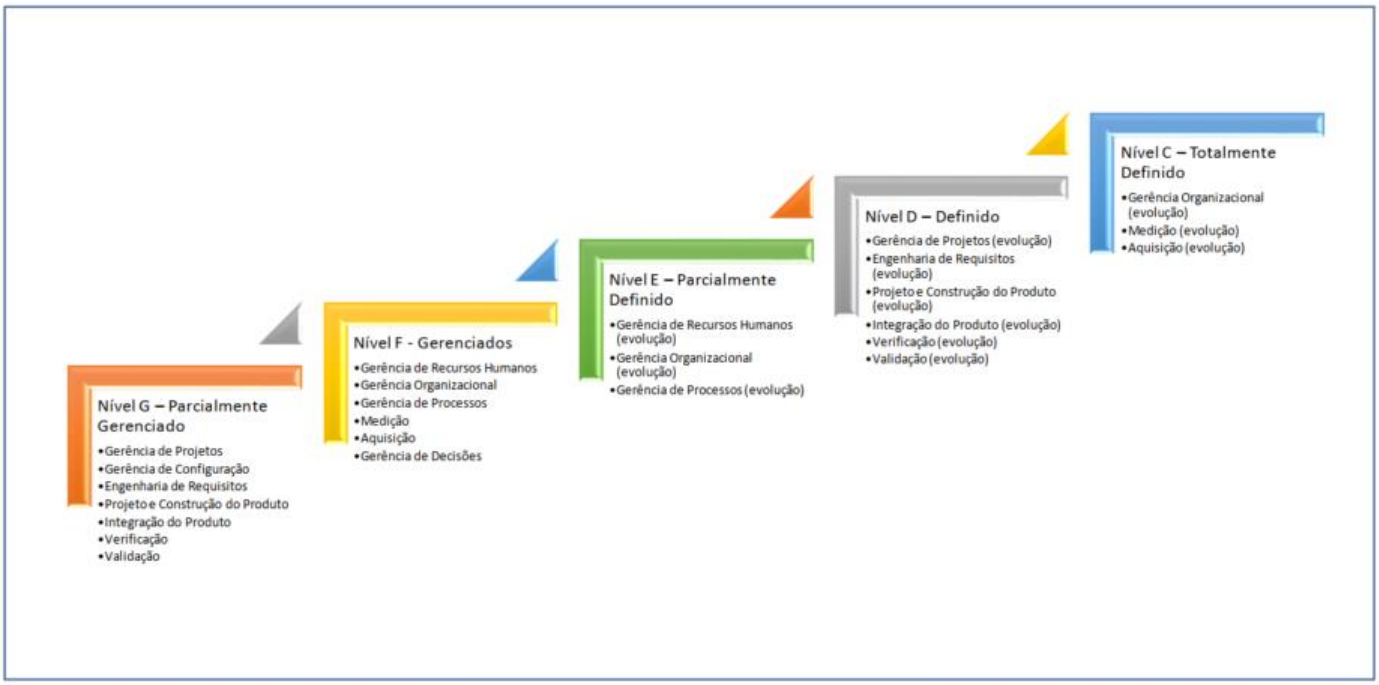
O Modelo de Referência MPS para Software (MR-MPS-SW) define níveis de maturidade que são uma combinação entre processos e sua capacidade.

Os processos estão divididos em 2 conjuntos:

- Os **processos de projeto** são aqueles que são executados para os projetos de software. Esses projetos podem ser de desenvolvimento de um novo produto, manutenção e evolução de produto.
- Os **processos organizacionais** são os processos concebidos para fornecer os recursos necessários para que o projeto atenda às expectativas e necessidades das partes interessadas da organização.



Os resultados esperados dos processos estão adequados a cada **nível de maturidade** pretendido, ou seja, nem todos os resultados estão presentes nos níveis iniciais e eles vão evoluindo à medida em que evolui a maturidade da organização. Os resultados são acumulativos, ou seja, os resultados que aparecem no nível G deverão estar presentes, com as mesmas características ou com evoluções, no nível F e acima.



As atividades e tarefas necessárias para atender ao propósito e aos resultados esperados não são definidas no Guia do MPSBR, devendo ficar a cargo dos usuários do MRMPS-SW identificarem as atividades e tarefas que atendam aos resultados esperados do processo e que sejam adequadas às suas características específicas.

A **capacidade do processo** é a caracterização da habilidade do processo para alcançar os objetivos de negócio atuais e futuros, estando relacionada com o atendimento aos atributos de processo (AP) associados aos processos de cada nível de maturidade. Os atributos de processos são específicos para cada conjunto de processos. Cada processo organizacional deve ser executado com os atributos de processo organizacionais pertinentes ao nível de maturidade. Os processos de projeto podem ser entendidos como um único processo que pode ser executado com os atributos de processos de projetos.

5.3.1. Capacidade do processo

A **capacidade do processo** é representada por um conjunto de atributos de processo. A **capacidade do processo** expressa o grau de refinamento e institucionalização com que o processo é executado na organização/unidade organizacional. O atendimento aos atributos de processo (AP) é requerido para todos os processos de acordo com a tabela a seguir:

Nível	Conjunto	Processo	Caracterização (AP)
G	Projeto	Gerência de Projetos	AP Nível 2 (Projeto)
		Gerência de Configuração	
		Engenharia de Requisitos	
		Projeto e Construção do Produto	
		Integração do Produto	
		Verificação	
		Validação	
F	Projeto	Gerência de Projetos	AP Nível 2 (Projeto)
		Gerência de Configuração	
		Engenharia de Requisitos	
		Projeto e Construção do Produto	
		Integração do Produto	
		Verificação	
		Validação	
	Organizacional	Gerência de Recursos Humanos	AP Nível 2 (Organizacional)
		Gerência Organizacional	
		Gerência de Processos	
		Medição	
		Aquisição	
		Gerência de Decisões	
E	Projeto	Gerência de Projetos	AP Nível 2 (Projeto)
		Gerência de Configuração	
		Engenharia de Requisitos	
		Projeto e Construção do Produto	
		Integração do Produto	
		Verificação	
		Validação	
	Organizacional	Medição	AP Nível 2 (Organizacional)
		Aquisição	
		Gerência de Recursos Humanos (evolução)	AP Nível 3 (Organizacional)
		Gerência Organizacional (evolução)	
		Gerência de Processos (evolução)	
		Gerência de Decisões	
D	Projeto	Gerência de Projetos (evolução)	AP Nível 3 (Projeto)
		Gerência de Configuração	
		Engenharia de Requisitos (evolução)	
		Projeto e Construção do Produto (evolução)	
		Integração do Produto (evolução)	
		Verificação (evolução)	
		Validação (evolução)	
	Organizacional	Medição	AP Nível 2 (Organizacional)
		Aquisição	
		Gerência de Recursos Humanos	AP Nível 3 (Organizacional)
		Gerência Organizacional	
		Gerência de Processos	
		Gerência de Decisões	
C	Projeto	Gerência de Projetos	AP Nível 3 (Projeto)
		Gerência de Configuração	
		Engenharia de Requisitos	
		Projeto e Construção do Produto	
		Integração do Produto	
		Verificação	
		Validação	
	Organizacional	Gerência de Recursos Humanos	AP Nível 3 (Organizacional)
		Gerência Organizacional (evolução)	
		Gerência de Processos	
		Medição (evolução)	
		Aquisição (evolução)	
		Gerência de Decisões	

Os diferentes níveis de capacidade dos processos são descritos por atributos de processo (AP). O alcance de cada atributo de processo é avaliado utilizando os respectivos resultados da implementação completa do atributo.

Atributo de Processo Nível 2 - A execução do processo é gerenciada: exprime a medida do quanto a execução do processo e os produtos de trabalhos são gerenciados.

- **Atributo de Processo Nível 2: Processos de Projetos**: como resultado da implementação completa deste atributo de processo:
  - (i) O processo produz os resultados definidos;
  - (ii) Um processo para o projeto é descrito, mantido atualizado e disponibilizado;
  - (iii) As pessoas estão preparadas para executar suas responsabilidades no processo;
  - (iv) A verificação de que o processo é seguido é realizada;
  - (v) Os produtos de trabalho selecionados são avaliados objetivamente ao longo do projeto em relação ao processo e padrões aplicáveis, os resultados são registrados, comunicados e a resolução de não conformidades é assegurada.
- **Atributo de Processo Nível 2: Processos Organizacionais**: como resultado da implementação completa deste atributo de processo:
  - (i) O processo produz os resultados definidos;
  - (ii) Um processo é definido, mantido e disponibilizado;
  - (iii) Os responsáveis pela execução do processo são definidos e comunicados;
  - (vi) As pessoas estão preparadas para executar suas responsabilidades no processo;
  - (iv) A verificação de que o processo é seguido é realizada.

Atributo de Processo Nível 3 – O processo é definido: exprime a medida do quanto o processo padrão da organização é mantido e implementado de forma a apoiar sua adaptação para um processo definido.

- **Atributo de Processo Nível 3: Processos de Projetos**: como resultado da implementação completa deste atributo de processo:
  - (i) O processo produz os resultados definidos;
  - (ii) Um processo padrão para projetos é estabelecido e inclui diretrizes para a sua adaptação a situações especificas;
  - (iii) As pessoas estão preparadas para executar suas responsabilidades no processo;

- (iv) É realizada verificação da aderência e efetividade do processo; (v) Os produtos de trabalho selecionados são avaliados objetivamente ao longo do projeto em relação ao processo e padrões aplicáveis, os resultados são registrados, comunicados e a resolução de não conformidades é assegurada;
- (vi) Oportunidades de melhoria no processo são identificadas durante as atividades de garantia da qualidade e a partir de resultados efetivos provenientes de análise de causa-raiz;
- (vii) Informações relacionadas ao processo ou ativos de processo são disponibilizadas para a organização.

- **Atributo de Processo Nível 3: Processos Organizacionais**: como resultado da implementação completa deste atributo de processo:
  - (i) O processo produz os resultados definidos;
  - (ii) Um processo padrão é estabelecido, mantido atualizado e disponibilizado, e inclui diretrizes para a sua adaptação a situações específicas;
  - (iii) O processo padrão e as diretrizes para adaptação são usados para planejar, executar e gerenciar o trabalho. O planejamento inclui identificação de papéis, responsabilidades, recursos e infraestrutura;
  - (iv) As pessoas estão preparadas para executar suas responsabilidades no processo;
  - (v) É realizada verificação da aderência e efetividade do processo;
  - (vi) Informações relacionadas ao processo ou ativos de processo são disponibilizadas para a organização.

5.3.2. Exclusão de processo

Alguns processos podem ser excluídos, total ou parcialmente, do escopo de uma avaliação MPS por não serem pertinentes ao negócio da unidade organizacional que está sendo avaliada. Cada exclusão deve ser justificada no Plano de Avaliação. A aceitação das exclusões e suas justificativas é responsabilidade do Avaliador Líder, conforme descrito no Guia de Avaliação.

5.4. Descrição dos processos do MR-MPS-SW

Vamos descrever os processos do MPS-SW em termos de seus propósitos e níveis de maturidade em que iniciam e passam por evolução:

5.4.1. Processos de Projeto

- **Gerência de Projetos – GPR (inicia no nível G e evolui no nível D)**: propósito de estabelecer e manter atualizados planos que definam as atividades, recursos, riscos, prazos e responsabilidades do projeto. Também é propósito deste processo prover informações sobre o andamento do projeto que permitam a realização de correções quando houver desvios significativos no desempenho do projeto, incluindo análise de causa-raiz.
- **Gerência de Configuração – GCO (inicia no nível G)**: propósito de estabelecer e manter a integridade de todos os produtos de trabalho de um projeto e disponibilizá-los a todos os envolvidos.
- **Engenharia de Requisitos – REQ (inicia no nível G e evolui no nível D)**: propósito de definir, gerenciar e manter atualizado os requisitos das partes interessadas e do produto, garantindo que inconsistências entre os requisitos, os planos e os produtos de trabalho sejam identificadas.
- **Projeto e Construção do Produto – PCP (inicia no nível G e evolui no nível D)**: propósito de projetar, desenvolver e implementar soluções para atender aos requisitos.
- **Integração do Produto – ITP (inicia no nível G e evolui no nível D)**: propósito de montar os componentes do produto conforme a estratégia definida, produzindo um produto integrado consistente com seu projeto (design) e seus requisitos.
- **Verificação – VER (inicia no nível G e evolui no nível D)**: propósito de confirmar que os produtos de trabalho selecionados atendem aos requisitos especificados pela execução de testes e revisão por pares.
- **Validação – VAL (inicia no nível G e evolui no nível D)**: propósito de confirmar que um produto ou componente do produto atenderá a seu uso pretendido quando colocado no ambiente operacional.

5.4.2. Processos Organizacionais

- **Aquisição – AQU (inicia no nível F e evolui no nível C)**: propósito de gerenciar a aquisição de produtos que satisfaçam às necessidades expressas pelo adquirente.
- **Medição – MED (inicia no nível F e evolui no nível C)**: propósito de coletar, armazenar, analisar e relatar dados objetivos relacionados aos produtos desenvolvidos e aos processos implementados nos projetos, de forma a apoiar os objetivos organizacionais.
- **Gerência de Decisões – GDE (inicia no nível F)**: propósito de analisar possíveis decisões críticas usando um processo formal, com critérios estabelecidos, para avaliação das alternativas identificadas.
- **Gerência de Recursos Humanos – GRH (inicia no nível F e evolui no nível E)**: propósito de prover a organização e os projetos com os recursos humanos necessários e manter suas competências adequadas às necessidades do negócio.
- **Gerência de Processos - GPC (inicia no nível F e evolui no nível E)**: propósito de estabelecer, manter atualizado, identificar e realizar melhorias em um conjunto de ativos de processo organizacional e padrões do ambiente de trabalho usáveis e aplicáveis às necessidades de negócio da organização. Também é propósito deste processo a definição da estratégia para garantia da qualidade nos projetos e da infraestrutura para realização de medições.
- **Gerência Organizacional - ORG (inicia no nível F e evolui no nível E e no nível C)**: propósito de fornecer para a gerência da organização instrumentos para apoiar os processos e prover um alinhamento entre os objetivos de negócio, os processos, os recursos e os projetos da organização.

Em resumo temos que:

Nível de maturidade	Processos do Nível de Maturidade
C	**Aquisição – AQU (evolução)** **Medição – MED (evolução)** **Gerência Organizacional – ORG (2ª evolução)**
D	**Gerência de Projetos – GPR (evolução)** **Engenharia de Requisitos - REQ (evolução)** **Projeto e Construção do Produto – PCP (evolução)** **Integração do Produto – ITP (evolução)** **Verificação – VER (evolução)** **Validação – VAL (evolução)**
E	**Gerência de Recursos Humanos – GRH (evolução)** **Gerência de Processos – GPC (evolução)** **Gerência Organizacional – ORG (1ª evolução)**
F	Aquisição – AQU Medição – MED Gerência de Decisões – GDE Gerência de Recursos Humanos – GRH Gerência de Processos – GPC Gerência Organizacional - ORG
G	Gerência de Projetos – GPR Gerência de Configuração – GCO Engenharia de Requisitos – REQ Projeto e Construção do Produto – PCP Integração do Produto – ITP Verificação – VER Validação – VAL

REFERÊNCIAS

SOFTEX. **Guia Geral MPS de Software**. 2020.