**Garantia e Qualidade de Software**

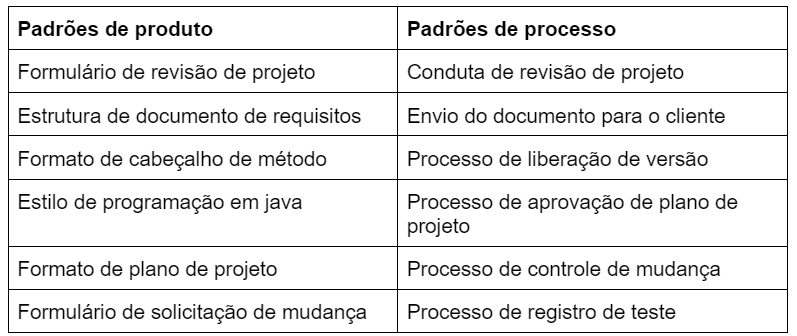
**O que considerar como Qualidade?**

Sommerville diz que a garantia de qualidade é o processo que define como a qualidade de software pode ser atingida e como a organização de desenvolvimento sabe que o software possui o nível de qualidade necessário.

Neste sentido, dois tipos de padrões podem ser definidos nesse processo:

**Padrões de produto:** padrões de documentos, tais como: estrutura de documentos de requisitos; padrões de assuntos incluídos na documentação; padrões de codificação para uso nas linguagens de programação.

**Padrões de processo:**tem relação com a definição dos processos de especificação, projeto e validação.



Os padrões de software são importantes, porque:

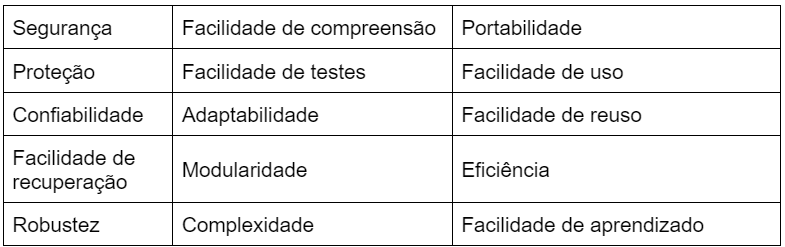
1. Evitam a repetição de erros cometidos em projetos anteriores, já que os padrões são definidos e baseados em experiências próprias da empresa.

2. Apresentam um framework com as melhores práticas, garantindo padrões, de acordo com as necessidades das empresas e a sua utilização.

3. Colaboram com a continuidade do sistema, facilitando a equipe que assumirá manutenções do sistema.

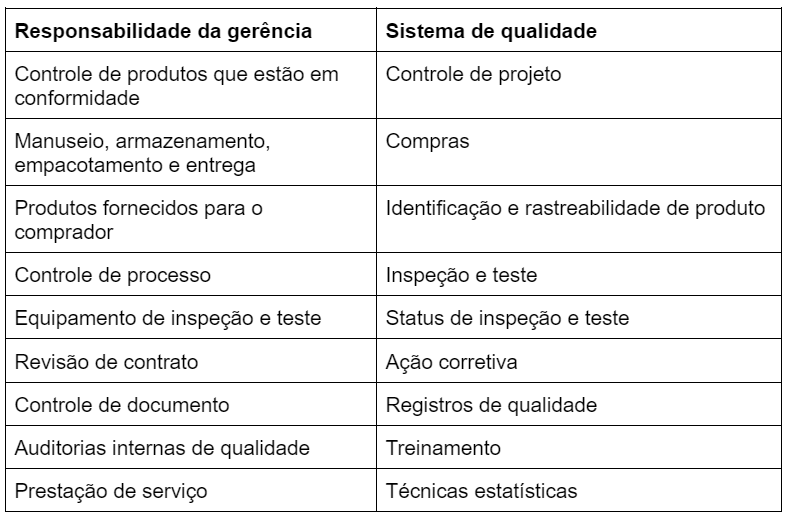
**Técnicas e Tipos de Validação**

Sommerville define alguns atributos de qualidade de software que precisam ser considerados no planejamento de qualidade:



O padrão **ISO 9001** se apresenta como um padrão geral para as empresas, provendo flexibilidade, não especificamente para desenvolvimento de softwares, mas possui princípios gerais que podem ser aplicados a esse fim.

Sommerville apresenta as áreas cobertas pelo modelo ISO 9001 para garantia de software:

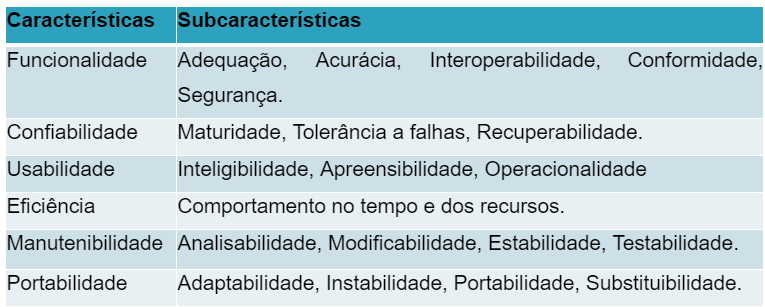


**ISO 9126** - ( *Information Technology - Software product evaluation -  
Quality characteristics and guidelines for their use*)

● Características de Qualidade para Domínios Específicos;

● Características de Qualidade para Tecnologias Específicas;

Definem seis características de qualidade e subcaracterísticas associadas a elas:

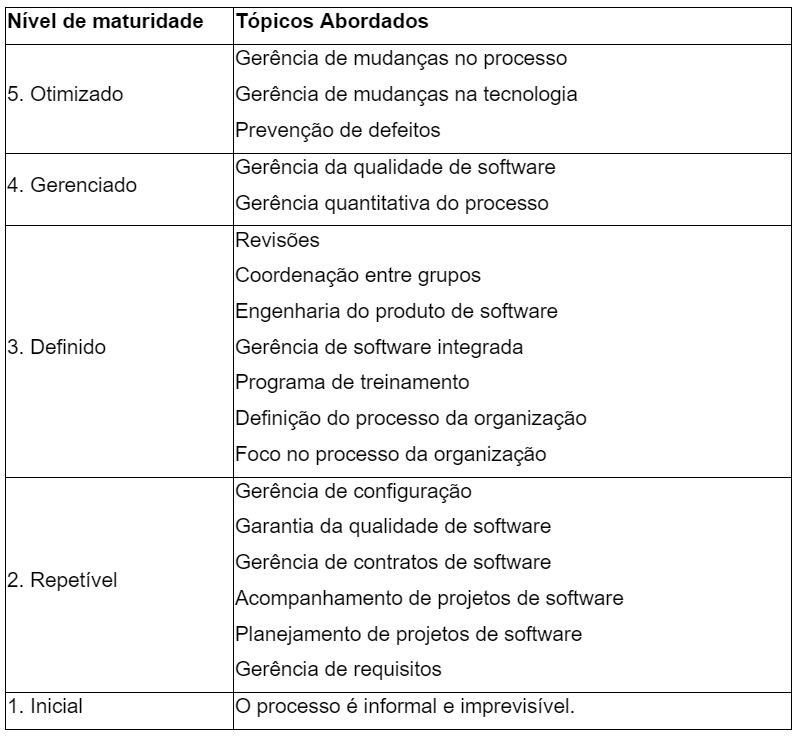


**CMM - Capability Maturity Model**

● Definido no Software Engineering Institute (SEI) - Carnegie Mellon University

● Motivação: Projetos do Departamento de Defesa

● 5 Níveis de Maturidade para o Processo



**TMM - Test Maturity Model**

A criação de um processo independente de teste demandou algumas necessidades de metodologias, métricas e melhorias. É um modelo de avaliação da maturidade de processo de teste e tem como objetivo dar suporte às organizações na melhoria do processo e tomou como base os seguintes paradigmas:

✔    É um modelo complementar ao CMM, com o qual mantém compatibilidade;

✔    É uma linha para a melhoria contínua do processo de testes;

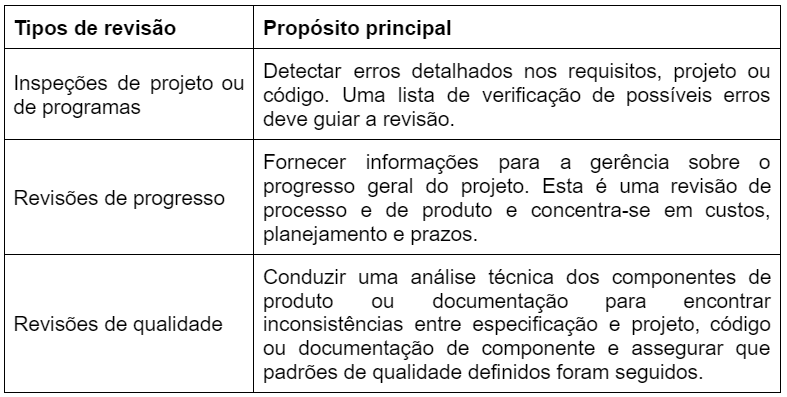
✔    É baseado na avaliação da situação atual do processo de testes através de regras claras e objetivas;

✔    É um modelo baseado nas melhores práticas de teste existentes no mercado.

Possui como níveis de avaliação: inicial, fase de definição, integração e otimização, prevenção de defeitos e controle de qualidade.

**Controle da qualidade de software**

O controle da qualidade de software é o conjunto planejado e sistemático de todas as ações necessárias para fornecer uma confiança adequada de que o item ou produto está de acordo com os requisitos técnicos estabelecidos.



A detecção de erros e/ou inconsistências no processo ou produto deve ser submetida à equipe para os ajustes necessários.

A avaliação de qualidade de um software é definida a partir da medição da qualidade. Para isso é necessário considerar três procedimentos:

1. **Definir o que medir**:  identificar dentre os componentes de sistemas os pontos-chaves do negócio, mais complexos e representativos.

2. **Definir as métricas**: definir a métrica a ser aplicada nos componentes. Uma métrica é a forma como se pode medir o comportamento do software. Podemos citar: Fan-in/Fan-out, extensão de código, complexidade ciclomática, extensão de identificadores, profundidade de aninhamento de declarações condicionais, índice de fog.

3. **Definir indicadores de qualidade**: é preciso definir para a situação de negócio qual resultado da métrica seria avaliado como médio, bom ou ótimo. Todo componente avaliado tem suas próprias características de atuação no ambiente, alterando a percepção na avaliação.

4. **Aplicar métricas e analisar resultados**: executar as métricas nos componentes selecionados e analisar os resultados em função dos indicadores.

**Exemplos de Validações**

**Revisão de requisitos**

É uma técnica que consiste em analisar e revisar sistematicamente todos os requisitos elicitados, executando uma checagem no tocante a erros e inconsistências.

Uma boa prática seria uma reunião formal com representantes ou especialistas de todas as áreas, tanto do contratante como do contratado.

Todas as equipes deverão ter representação.

E faz-se as seguintes atividades:

Preparo (antes do evento):

● Planejamento do que será revisado.

●  Estabelecer e convidar os envolvidos.

●  Definir local e tempo para a reunião.

●  Escolher para condução alguém “livre de vícios”, ou seja, que não estava integrado à equipe que desenvolveu o documento de requisito.

●  Distribuir previamente todos os documentos a serem utilizados na reunião.

●  Análise manual sistemática dos requisitos.

**Prototipação**

O protótipo é uma técnica de validação que tem como objetivo desenvolver uma simulação dos procedimentos para que os stakeholders possam concretizar as ideias propostas, minimizando inconsistências e problemas nos requisitos. “Nessa abordagem para avaliação, um modelo executável do sistema é demonstrado para os usuários finais e clientes.” (Sommerville, 2007).

Algumas preocupações podem ser consideradas na técnica:

●  Os *stakeholders*podem não entender a proposta da técnica e considerar que a qualidade está comprometida, visto que o ambiente não corresponde ao do usuário.

●  Estar atento com a equipe de desenvolvedores para não produzir componentes de baixa qualidade.

●  Construção de protótipos requer tempo que pode comprometer a previsão do projeto.

**Geração de casos de teste**

Uma propriedade importante para cada requisito é o de ser testável. Um requisito não testável pode ser um indício de geradores de problemas.

Para cada requisito funcional deve ser possível definir um ou mais testes a serem realizados no sistema final para ser possível verificar se o sistema cumpre o requisito na íntegra. Caso tal propriedade não esteja presente, ou até mesmo se for muito difícil testá-lo; tal circunstância indica a necessidade de uma retificação.  Na realização dos testes, deve-se tomar nota das características observadas quanto aos requisitos em si (identificador, requisitos relacionados), e como daquelas relacionadas aos testes (descrição, problemas, comentários, recomendações, etc.).

●  Desenvolver testes para os requisitos a fim de verificar a testabilidade.

●  Análise automatizada da consistência

**Automação de teste**

A diferença entre testes e automação de testes, é que no primeiro você realiza a tarefa de testar, e no segundo você usa um software que imita a interação com a aplicação no que se refere ao teste tal qual um ser humano faria (Graham e Fewster,1999).

●  testes produzidos sem requerer intervenção manual. Embutidos dentro dos componentes.

●  São necessários profissionais especializados e tempo no desenvolvimento de procedimentos para a automação.

Algumas dificuldades são encontradas, tais como:

●  Mudança da cultura.

●  Instabilidade dos requisitos decorrente de inúmeras mudanças quando o desenvolvimento está avançado.

●  Custo alto → perder trabalho já feito.

●  Modelo de domínio ainda não é amplamente utilizado pelos desenvolvedores.