# ETAPAS DE IMPLEMENTAÇÃO E TESTES DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

# IMPLEMENTAÇÃO

Quais são as entregas da etapa “projeto” do processo de desenvolvimento de software? Modelos, ou seja, diagramas e especificações textuais que permitem a implementação, ou codificação do software. Realizando um paralelo com a Engenharia Civil, temos plantas baixas, projeto estrutural, hidráulico-sanitário, elétrico etc.; agora, podemos ir para o terreno construir a casa.

A etapa “implementação” do processo de desenvolvimento de software permite realizar a tradução dos modelos de projeto em código executável por meio do uso de uma ou mais linguagens de programação.

Temos aqui um desafio! Ainda não conhecemos todos os detalhes dos algoritmos que efetivamente resolvem o problema; para solucioná-lo, entram em cena os programadores.

Podemos destacar o impacto de muitos requisitos não funcionais nesta etapa, tais como linguagens de programação, frameworks de persistência, e.g., framework para o mapeamento objeto-relacional escrito na linguagem Java denominado *hibernate*, sistema de gerenciamento de banco de dados, requisitos de qualidade, e.g., confiabilidade, usabilidade etc., reutilização de componentes, entre outros.

Nesse contexto, podemos descrever a denominada “Crise do software” sintetizada pela afirmativa de Edsger Dijkstra, em 1972:

> **A maior causa da crise do software é que as máquinas tornaram-se várias ordens de magnitude mais potentes! Em termos diretos, enquanto não havia máquinas, programar não era um problema; quando tivemos computadores fracos, isso se tornou um problema pequeno, e, agora que temos computadores gigantescos, programar tornou-se um problema gigantesco.**
>
> PRESSMAN, 2016.

Importante destacar que o desenvolvimento tecnológico do hardware nos últimos anos permitiu o desenvolvimento de softwares cada vez mais complexos, tendo forte impacto na indústria de software. Como exemplo,

podemos apresentar a substituição do paradigma estruturado pelo paradigma orientado a objetos, baseado na programação orientada a objetos, que permite o reúso intensivo de especificação, bem como melhor manutenibilidade e, como consequência, o desenvolvimento de softwares mais complexos.

# QUALIDADE DE SOFTWARE

Um dos fatores que exerce influência negativa sobre um projeto é a complexidade, estando associada a uma característica bastante simples: o tamanho das especificações. Em programas de computador, o problema de complexidade é ainda mais grave, em razão das interações entre os diversos componentes do sistema.

A discussão sobre problemas de software *versus* complexidade não estará completa sem abordar o assunto qualidade. De forma simplificada, a qualidade está em conformidade com os requisitos, tendo como objetivo satisfazer o cliente por meio da aplicação da Engenharia de Software, nosso contexto.

A norma ISO 9126 identifica seis atributos fundamentais de qualidade para o software:

- **FUNCIONALIDADE**
  Satisfação dos requisitos funcionais.

- **CONFIABILIDADE**
  Tempo de disponibilidade do software.

- **USABILIDADE**
  Facilidade de uso.

- **EFICIÊNCIA**
  Otimização dos recursos do sistema.

- **FACILIDADE DE MANUTENÇÃO**
  Realização de correção no software.

- **PORTABILIDADE**
  Adequação a diferentes ambientes.

A qualidade possui duas dimensões fundamentais aplicáveis ao software: as dimensões da qualidade do processo e da qualidade do produto.

Considerando que estamos descrevendo a etapa de implementação do processo de software, vamos apresentar considerações relativas à qualidade do produto software, ou seja, aos testes que garantem a qualidade do produto software.

# TESTE DE SOFTWARE

A qualidade do produto software pode ser garantida através de sistemáticas aplicações de testes nos vários estágios do desenvolvimento da aplicação. Esses testes permitem a validação da estrutura interna do software e sua aderência aos respectivos requisitos. A integração dos subsistemas também é avaliada, com destaque para as interfaces de comunicação existentes entre os referidos componentes de software.

Podemos, então, definir teste como um processo sistemático e planejado que tem por finalidade única a identificação de erros. No caso de softwares complexos, sabe-se que o teste será capaz de descobrir a presença de erros, mas não a sua ausência!

Softwares mal testados podem gerar prejuízos às empresas. Um defeito de projeto poderá encadear requisições de compras inadequadas, gerar resultados financeiros incorretos, entre outros. Até mesmo as tomadas de decisões gerenciais da empresa podem ser negativamente impactadas, pois muitos profissionais se apoiam nas informações com o objetivo de tornar a empresa mais eficiente. A qualidade das decisões está intimamente ligada à qualidade das informações disponibilizadas aos diversos níveis gerenciais.

Os procedimentos de testes aplicados diretamente em softwares são também conhecidos como testes de software ou testes dinâmicos, podendo, em sua maioria, sofrer alto nível de automação, o que possibilita a criação de complexos ambientes de testes que simulam diversos cenários de utilização.

## Atenção

Lembre-se: quanto mais cenários simulados, maior o nível de validação que obtemos do produto, caracterizando maior nível de qualidade do software desenvolvido.

Os testes de software têm por objetivo avaliar a qualidade desse produto nas mais variadas dimensões possíveis, ou seja, em relação à sua estrutura

interna, na aderência às regras de negócios estabelecidas, nos parâmetros de performance esperados pelo produto etc.

Durante a codificação do software, podemos adotar a estratégia representada na Figura 14. Essa espiral é percorrida a partir do interior, aumentando o nível de abstração a cada volta. Vamos descrever como cada etapa dessa estratégia funciona.

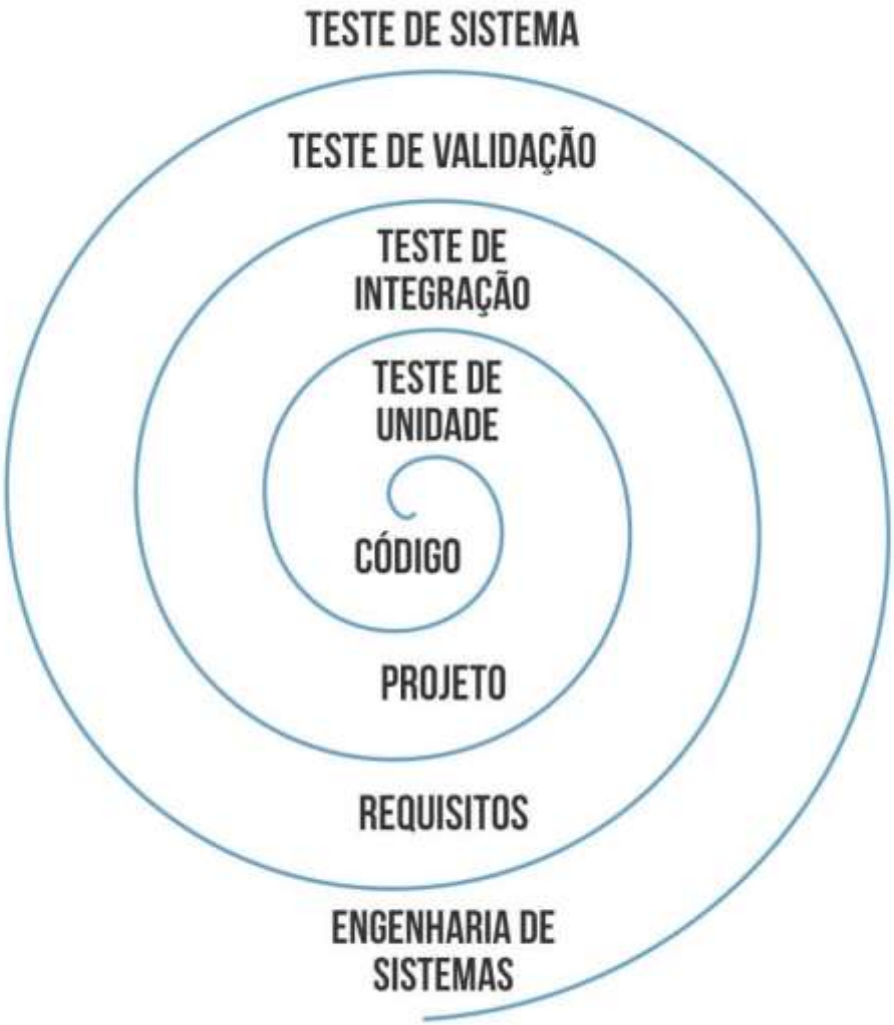


Figura 14 – Estratégia de Teste.

## TESTE DE UNIDADE

A validação da unidade de software é a primeira etapa da estratégia, tendo por objetivo testar componentes individuais de uma aplicação.
Nesta estratégia de testes, o objetivo é executar o software a fim de exercitar adequadamente toda a estrutura interna de um componente, como os desvios condicionais, os laços de processamento etc.

## TESTE DE INTEGRAÇÃO

Objetiva a manutenção de compatibilidade das novas unidades construídas, ou modificadas, com os componentes previamente existentes.
A compatibilidade de um componente é quebrada sempre que existe alteração ou exclusão de uma rotina ou propriedade pública de um componente. Quando isso ocorre, todos os outros componentes que empregam essas rotinas ou propriedades estão automaticamente incompatíveis com o componente modificado, gerando erros durante a execução do software. São exatamente esses erros que devem ser identificados nesse estágio dos testes de software.

## TESTE DE VALIDAÇÃO

Quando atingimos esse estágio dos testes, a maior parte das falhas de funcionalidade deve ter sido detectada nos testes unitários e nos testes de integração, sendo o objetivo desta etapa realizar a validação da solução como um todo. Os testes de validação contam com uma infraestrutura mais complexa de hardware, visando estar o mais próximo do ambiente real de produção.
Nesta etapa, são realizados os testes funcionais, que visam verificar se a versão corrente do sistema permite executar processos ou casos de uso completos do ponto de vista do usuário, de modo a obter os resultados esperados; e os testes não funcionais, que visam testar o software em relação às características não funcionais, como desempenho, segurança, confidencialidade, recuperação a falhas etc.

Devemos planejar os testes de software a partir dos cenários contidos nas descrições de casos de uso e identificados na etapa de análise do processo de software; normalmente, o engenheiro de software define os cenários com alto

nível de abstração lógica, denominados também de casos de teste. Neste caso, o analista de teste da equipe de qualidade deverá diminuir o nível de abstração dos referidos cenários, detalhando os mesmos por meio dos casos de testes. Como exemplo, um caso de uso “Realizar saque de conta”, onde podemos imaginar um cenário em que o saldo supera o valor do saque e a transação ocorre sem problema, ou cenários alternativos ou de exceção, tais como bloqueio do saque por saldo insuficiente; liberação do saque utilizando o limite disponível pelo banco com cobrança de juros; liberação do saque com retirada de fundo de aplicação, entre outros.

Nesta etapa, ocorrem os procedimentos de aceite, que deverão ser realizados a cada ciclo de implementação da solução, permitindo correções antecipadas de determinados pontos que não foram adequadamente atendidos pelo software.

Como essa é a última oportunidade efetiva de detectar falhas no software, poderemos empregar o aceite como uma estratégia para reduzir riscos de uma implantação onde um erro vital não detectado pode comprometer a imagem da solução como um todo. Dessa forma, podemos dividir o aceite em três momentos distintos: o Teste Alpha, o Teste Beta e o Aceite Formal.

### Aceite Teste Alpha

Os usuários finais são convidados a operar o software dentro de um ambiente controlado, sendo realizado na instalação do desenvolvedor.

### Aceite Teste Beta

Os usuários finais são convidados a operar o produto utilizando suas próprias instalações físicas, ou seja, o software é testado em um ambiente não controlado pelo desenvolvedor.

### Aceite Formal

Trata-se de uma variação do Teste Beta, cabendo aos próprios clientes e usuários determinarem o que deverá ser testado e validar se os requisitos foram adequadamente implementados.

A validação tem sucesso quando o software atende os requisitos, ou seja, funciona da forma esperada pelo usuário.

# TESTE DE SISTEMA

A última etapa na espiral, apresentada na Figura 14, é o teste de sistema, que extrapola os limites da engenharia de software, ou seja, considera o contexto mais amplo da engenharia de sistemas de computadores. Após ser validado, o software deve ser combinado com outros elementos do sistema, tais como hardware, base de dados etc.

Vejamos alguns testes sugeridos nesta etapa.

## Teste de recuperação

Este teste tem por objetivo avaliar o comportamento do software após a ocorrência de um erro ou de determinadas condições anormais.

Os testes de recuperação devem também contemplar os procedimentos de recuperação do estado inicial da transação interrompida, impedindo que determinados processamentos sejam realizados pela metade e sejam futuramente interpretados como completos. Como exemplo, simular saque com defeito no caixa eletrônico ou simular saque com queda de energia.

## Teste de segurança

Teste cujo objetivo é detectar as falhas de segurança que podem comprometer o sigilo e a fidelidade das informações, bem como provocar perdas de dados ou interrupções de processamento. Esses ataques à segurança do software podem ter origens internas ou externas.

A ideia é avaliar o nível de segurança que toda a infraestrutura oferece, simulando situações que provocam a quebra de protocolos de segurança. Como exemplo, avaliar se a senha do cartão está sendo requisitada antes e depois da transação ou avaliar se a senha adicional e randômica está sendo requisitada no início da operação.

## Teste por esforço

Teste com o objetivo de simular condições atípicas de utilização do software, provocando aumentos e reduções sucessivas de transações que superem os volumes máximos previstos para o software, gerando contínuas situações de pico e avaliando como o software e toda a infraestrutura estão se comportando. Como exemplo, simular 10.000 saques simultâneos.

## Teste de desempenho

Teste cujo objetivo é determinar se o desempenho, nas situações previstas de pico máximo de acesso e concorrência, está consistente com os requisitos definidos.

Devemos especificar os tempos de resposta considerados factíveis à realização de cada cenário. Como exemplo, garantir que manipulação com dispositivos físicos no saque não ultrapassem 10 segundos da operação.

## Teste de disponibilização

Também chamado de teste de configuração, tem por objetivo executar o software sobre diversas configurações de softwares e hardwares. A ideia é garantir que a solução tecnológica seja executada adequadamente sobre os mais variados ambientes de produção previstos nas fases de levantamento dos requisitos.

O referido teste também examina todos os procedimentos de instalação e instaladores a serem utilizados pelos clientes, bem como a documentação que será usada pelos usuários finais. Como exemplo, simular saque com impressora dos fornecedores X, Y e Z.

## Resumindo

Neste módulo, pudemos avaliar a importância das etapas de implementação e testes do processo de desenvolvimento de software.

À medida que o programador codifica a solução a partir dos modelos gerados na etapa de projeto, os testes são aplicados por meio de uma estratégia de testes em forma de espiral, incluindo quatro categorias de testes.

Inicialmente, o teste unitário permite validar a estrutura interna de cada componente de software.

O teste de integração possibilita validar a integração de novos componentes, ou componentes que passaram por manutenção, com os componentes previamente validados.

O teste de validação inclui a realização de testes de alto nível realizados pelos usuários finais do sistema, inicialmente em ambientes controlados pelo desenvolvedor e, posteriormente, em ambientes de produção não controlados pelo desenvolvedor, a fim de que ocorra a verificação do atendimento aos requisitos. Ao final, podemos ter a aceitação do sistema pelo usuário.

Finalizando, o teste de sistema, que extrapola os limites da Engenharia de Software, permite combinar o software com outros elementos do sistema, tais como hardware, base de dados etc.