Capítulo 1 - Fundamentos de Teste

1. Fundamentos de teste

1.1 O que é teste?

Os testes de software são essenciais para avaliar a qualidade de um sistema e reduzir o risco de falhas em operação. Problemas em softwares podem causar perda de dinheiro, tempo, reputação e até ferimentos ou mortes em casos críticos, como em hospitais ou aviões.

O teste de software não é só clicar e verificar resultados, mas envolve atividades como planejamento, análise, modelagem, execução e relatórios. Ele também verifica se os requisitos foram atendidos e se o software atende às necessidades reais dos usuários.

Além disso, o teste precisa estar alinhado ao ciclo de vida do desenvolvimento e ser adaptável. Embora existam ferramentas de automação, o teste é em grande parte uma atividade intelectual, que exige pensamento crítico, habilidades analíticas e atenção aos detalhes.

Teste e QA não são a mesma coisa. O teste é uma forma de controle de qualidade (QC).

> **Controle de Qualidade (QC)** : É uma abordagem **corretiva** e orientada para o **produto** que se concentra nas atividades que apoiam a obetações de **níveis adequados de qualidade.**
>
> **Quality Assurance (QA):** É uma abordagem **preventiva** e orientada a para o **processo,** se concentra na implementação e aprimoramento dos **processos** (se um bom processo for seguido corretamente, ele gerará um um bom produto).

1.1.1 Objetivos do teste

A seção aborda os principais objetivos do teste de software, com foco em como o teste contribui para a qualidade do produto e para a tomada de decisões dentro do desenvolvimento.

Objetivos principais do teste:

1. **Encontrar defeitos**
   Verificar se o sistema possui falhas ou comportamentos incorretos.

2. **Evitar defeitos**
   Prevenir problemas antes que eles sejam introduzidos, aplicando boas práticas e revisões desde as fases iniciais (como análise de requisitos).
3. **Criar confiança no nível de qualidade**
   Resultados positivos nos testes aumentam a confiança na estabilidade e qualidade do software.
4. **Fornecer informações para a tomada de decisão**
   Ajuda gestores a decidir se o sistema está pronto para ser liberado ou se precisa de correções.

Outros objetivos relevantes:

- Avaliar produtos de trabalho (como requisitos, código, planos de teste).
- Garantir cobertura necessária do objeto de teste.
- Reduzir riscos antes que o software chegue ao cliente.
- Verificar conformidade com requisitos.

1.1.2 Teste e depuração

Diferença entre Teste e Depuração:

Teste:

- Tem como objetivo **mostrar falhas** causadas por defeitos no software.
- Verifica se um defeito foi **realmente corrigido** (teste de **confirmação**).
- Garante que funcionalidades antigas **continuam funcionando** após mudanças (**teste de regressão**).
- É responsabilidade do **profissional de testes (QA)**.
- Testar **não corrige** o defeito — apenas o **detecta e relata**.

Depuração (Debug):

- É o **processo do desenvolvedor** de:
  - **Reproduzir** a falha relatada,
  - **Diagnosticar** a causa do problema,
  - **Corrigir** o defeito no código.
- Requer um **bom relatório de bug** do QA, com detalhes claros de como reproduzir o erro.

1.2 Por que os testes são necessários

Os testes de software são fundamentais para:

- **Atingir os objetivos do projeto** dentro de:
  - **Escopo**: Validando se o que foi pedido está sendo entregue corretamente.
  - **Tempo**: Evitando retrabalho futuro ao detectar problemas antes da entrega.
  - **Qualidade**: Melhorando o produto por meio de indicadores e estratégias de teste.
  - **Orçamento**: Utilizando técnicas eficazes que otimizam recursos financeiros e prazos.

Importância da colaboração:

- **Todos os stakeholders** (envolvidos no projeto) **podem e devem contribuir com testes**, cada um segundo suas habilidades:
  - **Desenvolvedores**: Testes técnicos (ex.: testes unitários).
  - **UX/designers**: Validação de layout, cores, usabilidade.
  - **Clientes/usuários finais**: Validam se o sistema atende às suas **necessidades reais**, mesmo sem conhecimento técnico.
  - **Testadores (QA)**: Especialistas que estruturam, executam e analisam testes de forma sistemática.

1.2.1 Contribuições do teste para o sucesso

1. Testes contribuem para sistemas de maior qualidade

- O teste **detecta defeitos**, que podem então ser **priorizados e corrigidos**.
- Isso **reduz o risco de problemas críticos em produção** e melhora a qualidade final do produto.
- Está diretamente ligado aos **objetivos principais do teste**, como vistos no item 1.1.1.

2. Teste avalia a qualidade em vários momentos do SDLC

- **SDLC (Software Development Life Cycle)**: ciclo de vida do desenvolvimento de software.
- O teste deve estar presente **desde a especificação até o aceite**, incluindo:
  - Testes de requisitos e documentos.
  - Testes a nível de código (ex: testes unitários).
  - Testes funcionais com o sistema rodando.
  - Teste de aceite (com o cliente).
- Ajuda em **decisões estratégicas**, como priorização de correções e releases.

3. Testes são essenciais para atender exigências contratuais, legais e regulatórias

- Quando há **contratos (ex: com o governo)**, os testes verificam se todos os **requisitos contratuais e legais** foram cumpridos.
- Também se aplicam a **normas específicas**, como normas contábeis, sanitárias, bancárias, etc.

4. Testadores devem entender as necessidades dos usuários

- O testador deve sempre **considerar o ponto de vista do usuário final**:
  - Compreender **por que** aquela funcionalidade existe.
  - Avaliar se **resolve bem o problema real** do usuário.
  - Identificar melhorias com base na **realidade do uso** (exemplo: sistema de posto de gasolina com botões intuitivos para frentistas).
- A empatia com o usuário **melhora a efetividade do teste** e agrega mais qualidade ao produto.

1.2.2 Relação entre Teste e Garantia da Qualidade (QA)

Teste ≠ Garantia da Qualidade (QA)

- Teste **não é sinônimo** de QA.
- O **teste** é uma **forma de controle da qualidade (QC – *Quality Control*)**, ou seja, é **uma parte do QA**.

Diferença entre QA e QC

QC – Controle da Qualidade:

- **Abordagem corretiva**, orientada ao **produto**.
- Foca em **verificar e corrigir defeitos** *após* eles existirem.
- Exemplo: testar o sistema, encontrar defeitos e encaminhar para correção.
- O **teste de software faz parte do QC**, pois verifica se o produto atende aos critérios definidos.

QA – Garantia da Qualidade:

- **Abordagem preventiva**, orientada ao **processo**.
- Foca em **evitar que defeitos ocorram**, melhorando os processos.
- Exemplo: implementar revisão de requisitos, validação de código com ferramentas como **SonarQube**, automações, validações de design etc.
- Um **bom processo** (QA) tende a gerar um **produto de boa qualidade**.

Quem é responsável pelo QA?

- **Todos os envolvidos** no projeto: desenvolvedores, analistas, designers, POs e, claro, testadores.
- **Testadores são especialistas**, mas atuam como **facilitadores e influenciadores** da qualidade nos outros times.
- Muitas vezes são os **impulsionadores de práticas e processos de QA**.

Resultados dos testes servem tanto para QA quanto para QC

- **QC (produto):** os resultados dos testes mostram **defeitos concretos** que precisam ser corrigidos.
- **QA (processo):** os resultados servem como **feedback do processo de desenvolvimento**.
  - Muitos defeitos? Pode indicar **problemas no processo**.
  - Poucos defeitos? Pode indicar **boa maturidade e eficiência do processo**.

1.2.3 Erros, Defeitos, Falhas e Causas-Raiz

→ **Erro**: O ser humano está sujeito a cometer um erro (engano)

→ **Defeito**: Que produz um defeito(bug) no código ou documento.

→ **Falha**: Se um defeito é executado, o sistema falha.

Causas-Raiz

Motivo fundamental para ocorrência de um problema

Analisando e remediamento a causa-raiz, é provável que outras falhas ou defeitos semelhantes

sejam evitados futuramente

1.3 Os Sete Princípios do Teste

Princípio 1 - O teste mostra a presença, não a ausência de defeitos

- O teste pode demonstrar a presença de defeitos, mas não pode provar que eles não existem.

Princípio 2 - Testes exaustivos são impossíveis

- Testar tudo não é viável.

Princípio 3 - Teste antecipados economizam tempo e dinheiro

- A atividade de teste deve começar o mais breve possível no ciclo de desenvolvimento.

Princípio 4 - Defeitos se agrupam

- Um número pequeno de funções contém a maioria dos defeitos.

Princípio 5 - Os testes se degradam

- Pode ocorrer de um mesmo conjunto de testes que são repetidos várias vezes não encontrarem novos defeitos após um determinando momento. Os cados de testes necessitam ser frequentemente revisados e atualizados

Princípio 6 - Os testes dependem do contexto

- Testes são realizados de forma diferente conforme o contexto.

Princípio 7 - Falácia da ausência de defeitos

- Encontrar defeitos não ajuda se o sistema construído não atende às expectativas dos usuários.

1.4 Atividades de teste, Testware e Papéis no teste

O processo de teste é um conjunto de atividades

- Assim como qualquer processo, o de teste é formado por **atividades organizadas e sequenciais**.
- Essas atividades visam aumentar a **probabilidade de sucesso** no cumprimento dos objetivos do teste.

Não existe processo de teste padrão

- **Não há um processo único ou universal** que sirva para todas as empresas ou tipos de software.
- Cada organização deve **adaptar seu processo** com base nas suas características e necessidades.

Processos precisam ser revisados e adaptados

- A busca pelo “processo ideal” é **contínua e dinâmica**.
- Sempre que surgir uma nova necessidade, ferramenta, problema ou oportunidade de melhoria, o processo deve ser **ajustado**.

O processo depende do contexto

- O **contexto** define como o processo de testes deve ser estruturado.

- Fatores como tipo de sistema, metodologia de desenvolvimento (ágil ou tradicional), orçamento, prazos, riscos e requisitos legais **influenciam diretamente** no desenho do processo de teste.

1.4.1 Atividades e Tarefas de Teste

→ Planejamento do Teste

Envolve atividades que **definem os objetivos e a abordagem do teste** para atender os objetivos do teste dentro das restrições impostas pelo contexto.

Os planos de testes podem ser **revisitados** com base no **feedback** das atividades de **monitoramento** e **controle.**

→ Monitoramento e Controle

Monitoramento: Verificação contínua de todas as atividades de teste e **compração** do progresso real com o plano de teste.

Controle: Tomada de ações necessárias para atender os objetivos do teste.

O progresso é comunicado aos Stakeholders por meio de relatórios

→ Análise do Teste - Determina “o que” testar

Analisa a **base de teste apropriada ao nível de teste** (especificações, modelagens, códigos, etc), para identificar **recursos testáveis** e definir e priorizar **condições de teste e riscos.**

A base de teste e os objetos de teste também são avaliados para identificar defeitos que possam conter e para avaliar sua testabilidade. Geralmente é apoiada pelo uso de técnicas de teste.

→ Modelagem do Teste - Determina “como” testar

Transforma condições de teste em casos de teste e outros materiais.

Identificação de itens de cobertura, que servem como guia para especificar as entradas do caso de teste.

Técnicas de teste podem ser usadas para apoiar a modelagem.

Também inclui definição dos requisitos de **dados de teste**, projeto do **ambiente de teste** e a identificação de qualquer outra infraestrutura e ferramenta necessária.

→ Implementação do Teste

- Criação ou aquisição do material de teste necessário para execução(ex. dados de teste).
- Casos de teste podem ser organizados em procedimentos de teste(passos) e geralmente são reunidos em conjuntos de teste.
- São criados scripts de teste manuais e automatizados.
- Procedimentos de teste são priorizados e organizados em um cronograma de execução de teste.
- O ambiente de teste é criado e verificado quanto à configuração correta

→ Execução do Teste

Executar/rodar testes de acordo com o cronograma de execução

Resultados atuais dos testes são comparados com os **resultados esperados**.

As anomalias(bugs) são analisadas para identificar suas causas prováveis . Essa análise permite relatar as anomalias com base nas falhas observadas.

→ Conclsuão do Teste

Ocorrem nos marcos do projeto(Ex: lançamento, fim da iteração, conclusão do nível de teste).

Qualquer materia de teste que possa ser útil no futuro é identificado e arquivado ou entregue às equipes apropriadas.

As atividades de teste são analisadas para identificar as lições aprendidas e as melhorias futuras.

Um relatório de conclusão do teste é criado e comunicado aos stakeholders.

1.4.2 Processo de Teste no Contexto

O processo de teste faz parte do desenvolvimento e deve atender às necessidades dos stakeholders.

Ele **não é fixo** e deve ser **adaptado ao contexto** do projeto, considerando fatores como:

- **Stakeholders:** O nível de envolvimento, as expectativas e a disposição para colaborar afetam diretamente o processo e as decisões relacionadas aos testes.
- **Equipe de projeto:** As habilidades, o conhecimento e a experiência dos membros da equipe, além da disponibilidade e necessidade de treinamento, impactam a forma como os testes serão conduzidos.
- **Domínio do negócio:** A complexidade da área de atuação da empresa, os riscos envolvidos e as exigências legais ou normativas influenciam diretamente os testes.
- **Fatores técnicos:** O tipo de software, sua arquitetura, tecnologias utilizadas e maturidade técnica afetam o planejamento e execução dos testes.
- **Restrições do projeto:** Escopo, tempo, orçamento e recursos disponíveis limitam o que pode ser feito nos testes e exigem decisões estratégicas.
- **Fatores organizacionais:** A estrutura da empresa, sua cultura, políticas internas e exigências externas influenciam os processos adotados para teste.
- **Ciclo de vida de desenvolvimento:** Metodologias como ágil ou tradicional impactam diretamente a abordagem e o momento em que os testes são realizados.
- **Ferramentas disponíveis:** A existência ou não de ferramentas adequadas para teste influencia o nível de automação, a cobertura e a forma de execução dos testes.

Esses fatores influenciam diversos aspectos do processo de teste, como:

- A estratégia de testes escolhida.
- As técnicas que serão aplicadas.
- O grau de automação possível.
- O nível de cobertura dos testes.
- O detalhamento da documentação.
- Os tipos de relatórios gerados e sua frequência.

Por fim, o principal ponto a ser lembrado é que o processo de teste deve sempre ser adaptado ao contexto em que está inserido.

1.4.3 Testware

Testware são os produtos de trabalho gerados durante as atividades de teste. Cada fase do processo de teste gera diferentes artefatos. Esses produtos variam conforme a empresa e o contexto.

Principais exemplos por atividade:

- **Planejamento:** plano de teste, cronograma, riscos, critérios de entrada/saída.
- **Monitoramento e controle:** relatórios de progresso, documentação de controle, info de riscos.
- **Análise:** condições de teste, priorização, relatórios de defeitos encontrados.
- **Modelagem:** casos de teste, cartas de teste (testes exploratórios), dados, cobertura, ambiente.
- **Implementação:** procedimentos de teste, scripts automatizados, dados, cronograma, setup.
- **Execução:** registros dos testes realizados e relatórios de defeitos.
- **Conclusão:** relatório final, lições aprendidas e planos de ação para melhorias.

Esses produtos devem ser organizados e versionados com apoio de ferramentas e gestão de configuração.

1.4.4 Rastreabilidade Entre a Base de Teste e o Testware

Rastreabilidade é a capacidade de **ligar artefatos de teste à sua origem** (como requisitos, especificações, riscos) e vice-versa.

Importância da rastreabilidade:

- Ajuda a entender de onde cada caso de teste veio e por que ele existe.
- Permite acompanhar a origem e os impactos de defeitos, mudanças e resultados de testes.
- Facilita o monitoramento da cobertura de requisitos, avaliação de riscos e identificação de lacunas.
- Torna auditorias, relatórios e o acompanhamento do progresso mais claros e confiáveis.
- Ajuda a avaliar qualidade do produto, eficácia do processo de desenvolvimento e cumprimento dos objetivos do projeto.

Exemplos:

- Rastrear casos de teste aos requisitos mostra se tudo está coberto.

- Rastrear resultados de testes aos riscos ajuda a avaliar o nível de exposição.
- Em mudanças, saber quais testes estão ligados ao item alterado facilita o planejamento da revalidação.

A rastreabilidade pode ser **manual ou feita por ferramentas**, e deve ser **bidirecional** (ida e volta).

1.4.5 Papéis no Teste

O *Syllabus* define dois papéis principais no teste:

1. Gerente de Teste

- Responsável pelo processo de teste, equipe e liderança das atividades.
- Atua principalmente no planejamento, monitoramento, controle e conclusão dos testes.
- Relaciona-se com os stakeholders, faz relatórios e define estratégias.
- No desenvolvimento ágil, parte dessas tarefas pode ser feita pela equipe de desenvolvimento ou por gerentes externos.

2. Testador

- Responsável pela parte operacional e técnica dos testes.
- Atua na análise, modelagem, implementação, execução de testes e avaliação de resultados.
- Pode lidar com testes funcionais e não funcionais (ex: desempenho, usabilidade, segurança).
- Pode ser exercido por qualquer pessoa com conhecimento do produto, incluindo cliente ou usuários, dependendo do contexto.

Observações importantes:

- Papéis podem variar conforme o projeto e o contexto.
- Uma mesma pessoa pode exercer ambos os papéis, dependendo da necessidade.
- Nem sempre haverá alguém com o cargo formal de gerente de testes, mas o papel ainda pode ser assumido por alguém da equipe.

1.5 Habilidades Essenciais e boas Práticas em testes

1.5.1 Habilidades Genéricas Necessárias para Testes

- **Conhecimento sobre testes:** fundamental para quem exerce o papel de testador profissional.

- **Atenção aos detalhes, curiosidade, meticulosidade:** ajudam a encontrar falhas que passariam despercebidas.
- **Boa comunicação:** essencial para relatar defeitos, participar de reuniões, influenciar o time e lidar com stakeholders.
- **Trabalho em equipe e escuta ativa:** entender necessidades e ajustar a estratégia de testes.
- **Pensamento analítico, crítico e criatividade:** fundamentais para modelar testes eficazes e cobrir caminhos alternativos.
- **Conhecimento técnico:** aumenta a eficiência, facilita a automação e o entendimento do sistema.
- **Conhecimento do domínio:** entender o negócio ajuda a testar com mais precisão.
- **Testadores como portadores de más notícias:** exige comunicação construtiva e empatia para evitar conflitos.
- **Viés de confirmação:** pode afetar a aceitação de defeitos; é importante manter postura neutra e objetiva.
- **Teste como atividade crítica:** defeitos devem ser comunicados sem atacar pessoas; foco no problema, não no autor.

1.5.2 Abordagem de equipe completa

- **Origem:** é uma prática do Extreme Programming (XP), uma metodologia ágil.
- **Definição:** qualquer membro da equipe, se tiver o conhecimento e as habilidades necessárias, pode realizar qualquer tarefa — mesmo fora da sua especialidade.
- **Exemplo:** um testador que entende de desenvolvimento pode programar algo se a equipe precisar. Um desenvolvedor pode ajudar nos testes se for necessário.
- **Responsabilidade compartilhada:** todos os membros da equipe são responsáveis pela **qualidade** do produto, não apenas os testadores.
- **Colaboração ativa:** testadores trabalham próximos:
  - De desenvolvedores, ajudando em decisões de automação e testes unitários.
  - De analistas ou representantes do negócio, ajudando a criar bons testes de aceite.
- **Ambiente de trabalho:** todos compartilham o mesmo espaço físico ou virtual (Slack, Teams, etc.), mantendo **comunicação constante e eficaz**.
- **Benefícios:**
  - Melhora a **dinâmica e a comunicação da equipe**.

    - Aumenta a **agilidade** e o **comprometimento com a qualidade**.
    - Permite que os testadores **compartilhem seu conhecimento de testes com toda a equipe**, influenciando o produto desde o início.
- **Limitação:** nem sempre essa abordagem é adequada.
    - **Sistemas críticos** (como aviação, saúde ou bancos) podem exigir **equipes de teste mais independentes**.
    - A proximidade da equipe pode reduzir a imparcialidade — o que é um risco em projetos onde a **neutralidade na validação é essencial**.

1.5.3 Independência dos testes

A independência dos testes significa que quem executa os testes não deve ser o mesmo que desenvolveu o código. Isso ajuda a evitar o viés de confirmação — quando o próprio desenvolvedor acredita que seu código está correto e testa de forma mais superficial. Quanto maior a distância entre quem desenvolve e quem testa, maior a independência.

Níveis de independência:

- **Sem independência:** desenvolvedor testa seu próprio código.
- **Alguma independência:** outro desenvolvedor ou testador dentro da mesma equipe testa.
- **Alta independência:** testadores fora da equipe de desenvolvimento, mas na mesma organização.
- **Muito alta independência:** testadores externos, contratados de fora da empresa.

Vantagens da independência:

- Testadores independentes tendem a encontrar tipos de falhas diferentes, pois têm uma visão externa e imparcial.
- Eles questionam mais as suposições feitas durante o desenvolvimento e especificação, ajudando a garantir maior qualidade.
- Por não terem apego emocional ao projeto, são menos tendenciosos no que deve ser testado.

Desvantagens da independência:

- Pode haver falta de colaboração entre testadores e desenvolvedores, o que dificulta a comunicação e pode atrasar o feedback.
- Testadores independentes podem ser vistos como um “gargalo” no processo, especialmente se os testes são realizados só no final do desenvolvimento.

- Desenvolvedores podem perder a sensação de responsabilidade pela qualidade do software, acreditando que “isso é problema do time de testes”.
- Testadores externos podem ter menos informações importantes sobre o projeto, dificultando o entendimento do sistema.

A independência nos testes traz imparcialidade e ajuda a detectar falhas que os desenvolvedores podem não ver. No entanto, ela pode prejudicar a comunicação e o senso de responsabilidade compartilhada. Por isso, é importante avaliar o contexto do projeto para decidir o nível ideal de independência entre desenvolvimento e testes.
