Defeito, Falha e Erro

Antes de abordar testes de software, é essencial diferenciar defeito, falha e erro:

- **Defeito**: ocorre quando há uma instrução ou comando incorreto. No contexto de um veículo, seria quando ele não responde corretamente ao motorista.
- **Falha**: manifesta-se como um comportamento inconsistente, permitindo que o sistema funcione de forma precária.
- **Erro**: é um desvio da especificação, como a montagem incorreta de um componente.

Testes são fundamentais para identificar esses problemas e garantir a qualidade do software. Assim como testes rigorosos comprovaram a segurança de veículos, no desenvolvimento de software, os testes estruturados reforçam a confiabilidade do produto final.

Teste de Software e Estratégia de Testes

De acordo com Pressman (2010), testar software permite identificar erros introduzidos durante o desenvolvimento. O processo deve ser sistemático, incluindo planos, estratégias e especificação de testes. A IBM (2006) destaca que uma estratégia de testes deve abranger:

- **Quem faz?** Gerentes de projeto, engenheiros de software e especialistas em testes.
- **Por que é importante?** Evita desperdício de tempo e reduz a chance de erros passarem despercebidos.
- **Quais são os passos?** O teste começa pelos componentes, avança para integração, validação e sistema.
- **Como garantir que os testes foram feitos corretamente?** Através de revisão e avaliação dos planos de teste.

Uma boa estratégia de testes deve ser flexível para se adaptar ao projeto, mas também estruturada para garantir planejamento e acompanhamento eficiente.

Tipos de Testes

Pressman (2010) e Sommerville (2007) definem diferentes estágios de testes:

- **Teste de Unidade:** Avalia módulos individuais do software (teste de caixa branca).
- **Teste de Integração:** Verifica a interação entre componentes.
- **Teste de Validação:** Confirma que o software atende aos requisitos do cliente.
- **Teste de Sistema:** Avalia o software como um todo, incluindo:
    - *Recuperação:* Testa a capacidade de recuperação após falhas.
    - *Segurança:* Avalia a resistência a ataques.
    - *Estresse:* Simula condições extremas de uso.
    - *Desempenho:* Mede a eficiência do sistema sob diferentes cargas.

Testes Automatizados

Bastos et al. (2007) afirmam que testes iniciados cedo reduzem custos de correção. O ciclo de vida dos testes inclui planejamento, preparação, especificação, execução e entrega.

Testes automatizados, segundo Bernardo e Kon (2008), permitem:

- Repetição rápida e eficiente de casos de teste.
- Execução de cenários complexos impossíveis manualmente.
- Simulação de centenas de usuários e grandes volumes de dados.

A abordagem de integração contínua, incentivada pelas metodologias ágeis, assegura que o código esteja sempre testado e confiável, reduzindo a necessidade de retrabalho.

Executar testes frequentemente é essencial para evitar acumulação de erros, garantir eficiência e evitar custos elevados de manutenção.

Testes não automatizados

Após a elaboração do software, a implementação de testes manuais ou automatizados torna-se essencial. Esse processo exige um conhecimento multidisciplinar, abrangendo programação, técnicas de teste de software, compreensão da linguagem de negócio e domínio de ferramentas específicas. Por essa razão, os profissionais responsáveis pela automação de testes possuem um perfil diferenciado, pois precisam equilibrar expertise técnica e visão estratégica para garantir a qualidade do sistema.

Segundo Schach (2007), a implementação consiste na conversão do projeto detalhado em código. Embora esse processo fosse mais simples se realizado por uma única pessoa, a complexidade dos sistemas modernos exige equipes de desenvolvimento trabalhando simultaneamente em diferentes componentes. Esse cenário torna a testabilidade mais desafiadora, demandando colaboração constante entre desenvolvedores e testadores para assegurar a confiabilidade e o desempenho do software.

A adoção de processos de qualidade na engenharia de software tem como objetivo garantir produtos mais robustos e previsíveis, seguindo padrões ao longo de todo o ciclo de vida do software. Esse processo, conhecido como ciclo de vida da qualidade de software, inicia-se na concepção do produto e continua até sua descontinuidade. Além de contribuir para o controle do desenvolvimento, ele auxilia na definição de prazos e na mitigação de riscos.

Modelos de qualidade, como CMMI e MPS.BR, estabelecem boas práticas e normatizam os processos de desenvolvimento, promovendo a criação de produtos de alto nível. Neste módulo, exploraremos esses conceitos e sua aplicação na engenharia de software.