Engenharia de Software

Testes de software

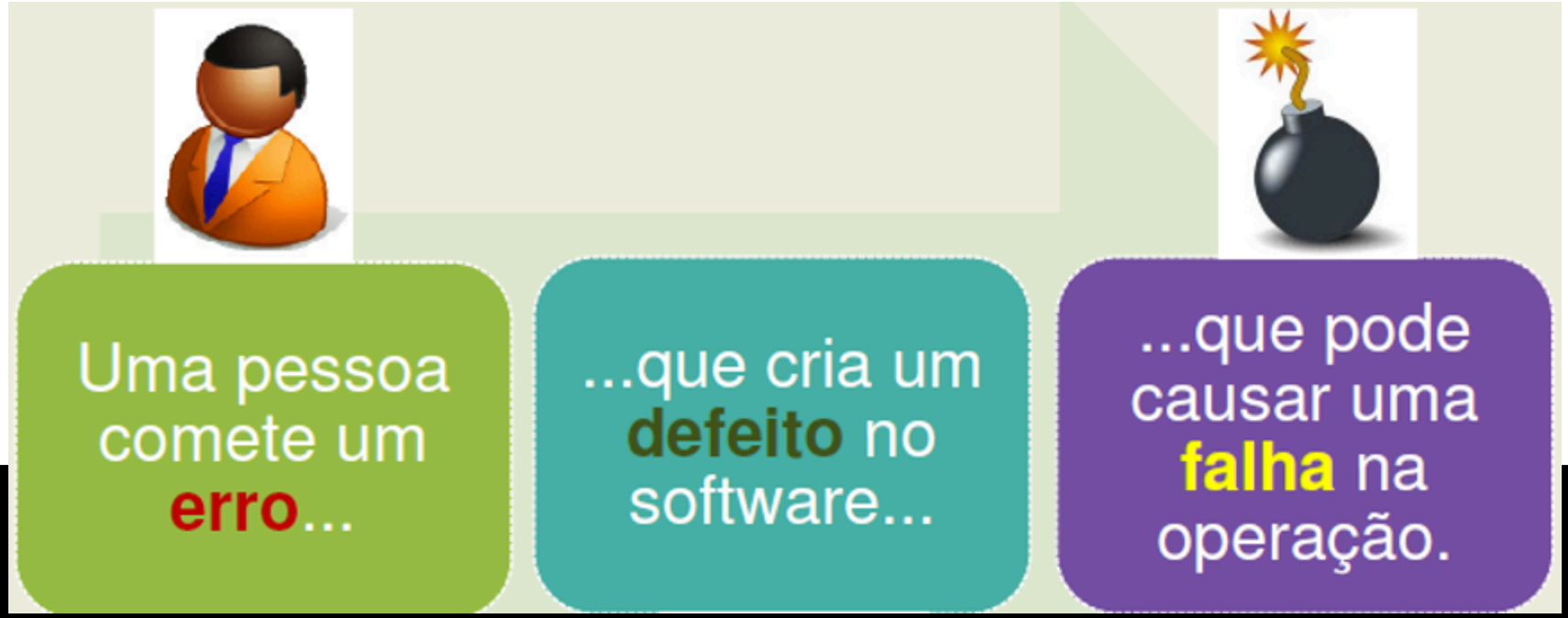
Marta Martinho 2020

Organização da apresentação

- Testes de software;
- O que é um teste de software?;
- Porque é que os testes de software são importantes?;
- Benefícios dos testes de software;
- Princípios dos testes de software;
- Atividades ligadas aos testes de software;
- Tipos de testes de software;
- Técnicas de testes de software;
- CI & CD - Continuous integration & Continuous delivery
- Refactoring

Testes de software

- Os testes de software são usados para verificar se o produto de software atende aos requisitos esperados e para garantir que o produto de software esteja livre de defeitos (bugs), evitando falhas;
- Tem como objetivo identificar bugs, lacunas ou requisitos ausentes comparados com os requisitos especificados;
- Os defeitos podem manifestar-se através de bugs, como resultados inesperados ou operações não realizadas (e.g. cálculos errados, dados não armazenados);



O que é um teste de software

"O teste consiste em executar o programa com a intenção de encontrar erros (*bugs*)".
[The Art of Software Testing – Glendford Myers, John Wiley & Son, 1979]

"Teste de software é o processo formal de avaliar um sistema ou componente de um sistema por meios manuais ou automáticos para
verificar se satisfaz os requisitos especificados ou identificar diferenças entre os resultados esperados e os obtidos"
[IEEE 729 - Glossary of Software Engineering Terminology, 1983]

Teste de software consiste na verificação dinâmica do comportamento de um programa, através de um conjunto finito de casos de teste, adequadamente selecionado a partir de um conjunto infinito de possibilidades,
contra um comportamento esperado especificado.
[SWEBOK - Guide to the Software Engineering Body of Knowledge]

O que é um teste de software

•Segundo Sommervile (2007), “[...] a meta do teste de software é convencer os desenvolvedores e clientes do sistema de que o software é bom o suficiente para o uso operacional. O teste é um processo voltado a atingir a confiabilidade do software”.

•Segundo Pressman (2011), a qualidade pode ser definida como: “uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que produzem e para aqueles que utilizam”.

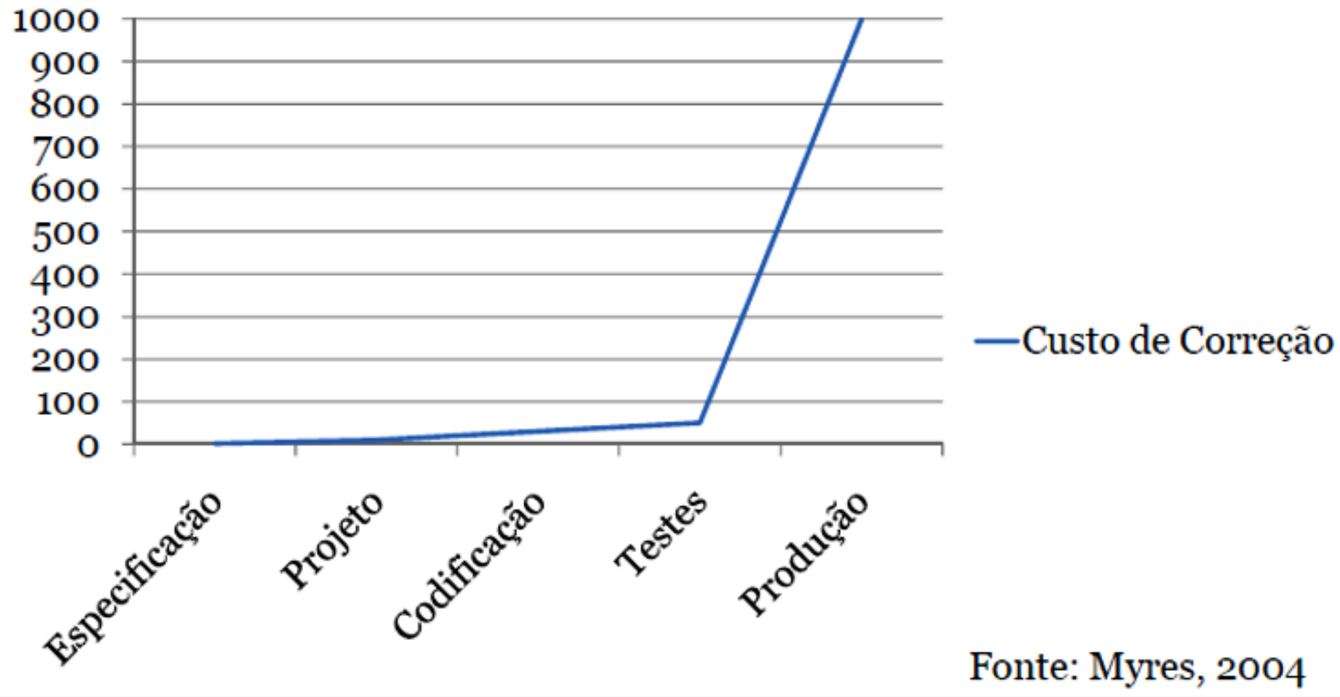
O que é um teste de software

- Segundo Myers (1979), o objetivo da fase de teste é o processo de executar um programa com a intenção de descobrir um erro através de um bom caso de teste.

- Segundo Pressman (2000), teste de software é um elemento crítico para a garantia da qualidade do produto e representa a ultima revisão de especificação, projeto e codificação.

Testes de software

- O custo da correção de um defeito tende a ser cada vez maior quanto mais tarde ele for descoberto [Myres, 2004]



Fonte: Myres, 2004

Porque é que os testes de software são importantes?

- A importância dos testes de software está no facto de, tendo a possibilidade de um bug ser identificado precocemente, poderá ser resolvido antes da entrada em produção do produto de software;
- Um produto de software devidamente testado garante confiabilidade, segurança e desempenho, resultando em menores gastos a nível de tempo, economia e mais satisfação por parte do cliente;
- Os bugs de software podem ser caros ou até mesmo perigosos, podendo causar perdas monetárias e humanas, estando a história repleta de exemplos:
    - ✓ A Starbucks foi forçada a fechar cerca de 60% das lojas nos EUA e Canadá devido a uma falha de software no seu sistema de POS. Durante um período de tempo a loja serviu café de graça.
    - ✓ Em abril de 1999, um bug de software causou a falha de lançamento de um satélite militar dos EUA de mais de mil milhões de dólares, o acidente mais caro da história.
    - ✓ Em maio de 1996, um bug de software fez com que as contas bancárias de 823 clientes de um grande banco dos EUA fossem creditadas com 920 milhões de dólares.
    - ✓ O Airbus A300 da China Airlines caiu devido a um bug de software em 26 de abril de 1994. Morreram 264 pessoas.

Benefícios dos testes de software

- **Custo-benefício:** é uma das vantagens importantes dos testes de software. Testar qualquer projeto de TI atempadamente ajuda a economizar dinheiro a longo prazo. No caso dos bugs serem detetados antes da fase de testes de software, custa menos a reparar;
- **Segurança:** é o benefício mais vulnerável e sensível dos testes de software, uma vez que os testes ajudam a remover riscos e problemas relacionados com segurança atempadamente. Os utilizadores procuram produtos que lhes transmitam confiança e segurança no uso.
- **Qualidade do produto:** É um requisito essencial de qualquer produto de software. Os testes garantem que um produto de qualidade seja entregue aos clientes.
- **Satisfação do Cliente:** O principal objetivo de qualquer produto é a satisfação do cliente. Os testes de UI garantem uma melhor experiência para o utilizador.

Principios dos testes de software

1. Os testes permitem mostrar a presença de defeitos;
2. Os testes exaustivos são impossíveis;
3. Os testes devem ser realizados logo que possível no ciclo de desenvolvimento de software;
4. Agrupar os defeitos. Onde e como testar?;
5. O paradoxo do pesticida;
6. Os testes dependem do contexto;
7. A ilusão da ausência de defeitos.

Principios dos testes de software

- **Os testes permitem mostrar a presença de defeitos.**

  ✓ Não provam a ausência de erros de conceção e/ou implementação;

  ✓ Não descobrir um defeito não prova a inexistência de defeitos.

- **Os testes exaustivos são impossíveis.**

  ✓ É impossível explorar a combinação dos cenários possíveis de forma exaustiva. Consideremos a soma de dois inteiros com 32 bits: cada operando pode ter 2^{32} valores distintos. A quantidade de casos distintos de execução é portanto igual a 2^{64}. Testar todos os casos, com 1000 milhões de operações por segundo, levaria 127 anos!.

Principios dos testes de software

- **Os testes devem ser realizados logo que possível no ciclo de desenvolvimento de software**.

  ✓ Os custos associados aos defeitos aumentam com o atraso no início dos testes, sendo mais difíceis de localizar, e envolvendo uma parte cada vez maior da aplicação. A dificuldade da correção dos defeitos aumenta com o tempo (aumento do custo e esforço necessário).

    ➡ Custo na fase de especificação 1

    ➡ Custo na fase de implementação 10

    ➡ Custo na fase de produção 100

  ✓ Os esforços na fase de manutenção podem ser reportado para a fase a conceção e desenvolvimento.

.

Principios dos testes de software

- **Agrupar os defeitos. Onde e como testar?**

  ✓ Primeira abordagem: distribuir o esforço dos testes de forma uniforme no conjunto da aplicação;

  ✓ No entanto grande parte dos defeitos concentram-se na pequena porção da aplicação (regra 80/20). 80% dos defeitos concentram-se em 20% da aplicação.

  ✓ Uso de modelos de defeitos para previsão das porções da aplicação com maior quantidade de defeitos

  ➡Componentes transacionais;

  ➡Componentes multitarefas;

  ➡Componentes de tempo real.

  ✓ Problemas com concorrência/sincronização.

Principios dos testes de software

- **O paradoxo do pesticida**

  ✓ Quando devemos terminar a realização de testes?

  ✓ Na maioria dos casos, a quantidade de defeitos descobertos reduz-se após um período de tempo de testes.

  ✓ Existe um momento a partir do qual um mesmo tipo de teste não permitirá descobrir novos defeitos.

  ✓ Aplicar em demasia uma técnica acaba por torna-la ineficaz.

  ✓ Necessidade de renovar os tipos de testes, alterando as perspetivas.

Principios dos testes de software

- **Os testes dependem do contexto**

  ✓ Algumas aplicações podem ser executadas em contextos muito diferentes.

  ✓ De acordo com o contexto, os objetivos e os tipos de testes podem não ser idênticos: testes funcionais, testes de desempenho, testes de segurança, etc.

Principios dos testes de software

- **A ilusão da ausência de defeitos**

  ✓ Os testes permitem descobrir defeitos tendo em consideração as espectativas teóricas da aplicação, podendo ser diferentes das espectativas reais.

  ✓ Para melhorar a qualidade de uma aplicação não é suficiente detetar e corrigir os defeitos.

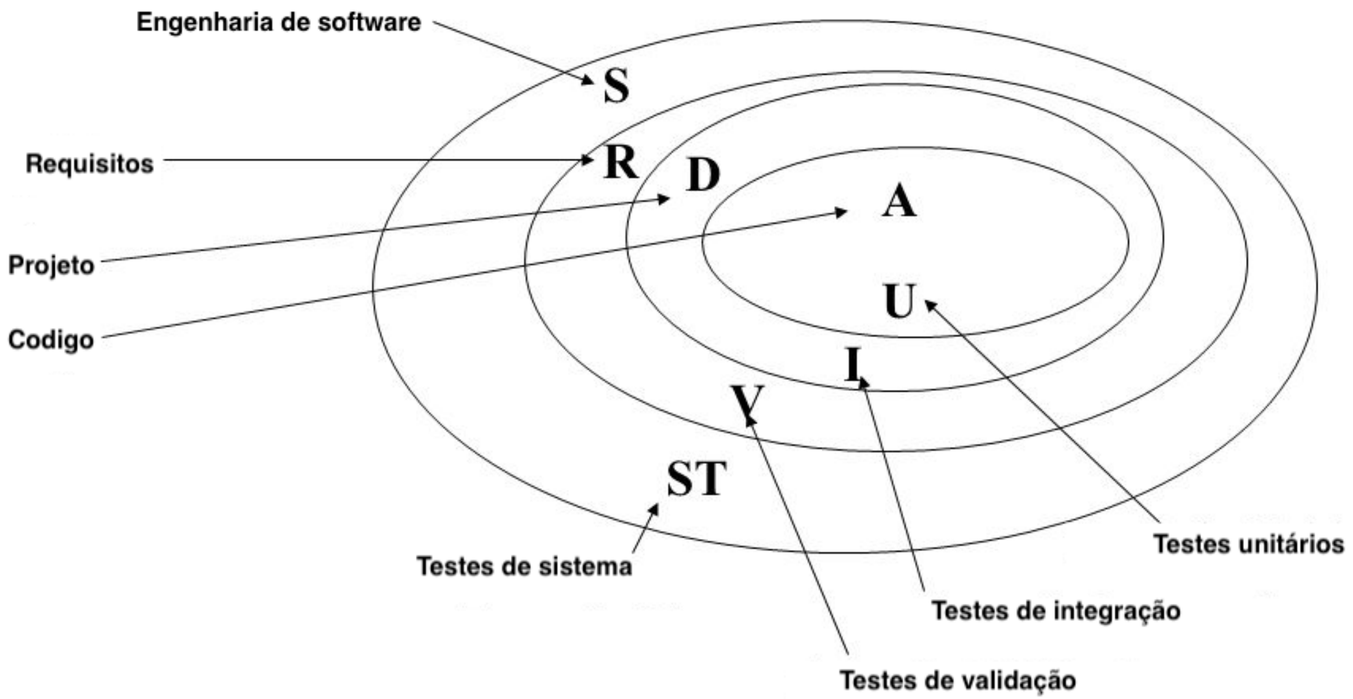
  ✓ Norma ISSO 9126 sobre qualidade (critérios principais funcionalidade, usabilidade, fiabilidade, portabilidade, eficiência, maintainability)

  ✓ A ausência de defeitos num critério não garante uma qualidade aceitável numa outra perspetiva.

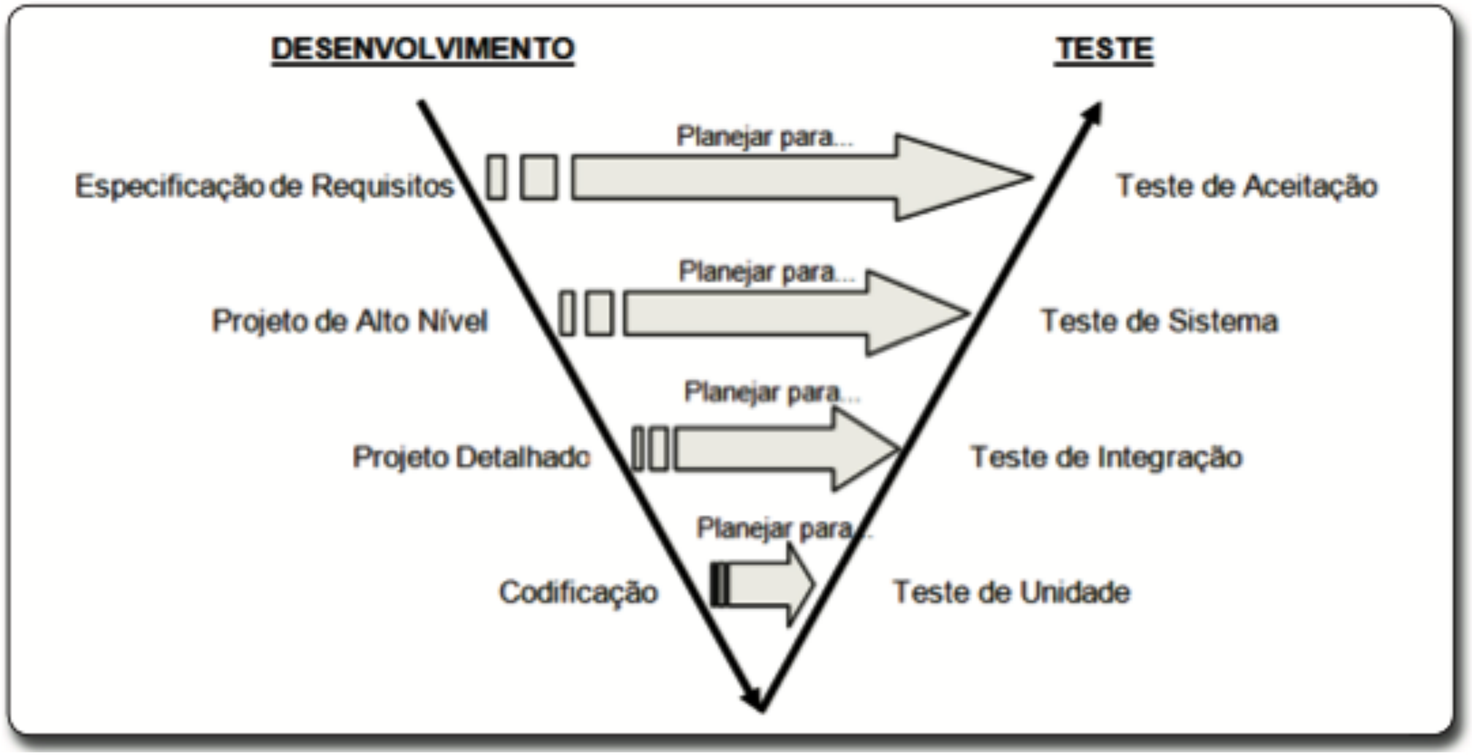
Tipo de testes de software

- **Planificar os testes**
    - ✓ Onde testar?
    - ✓ Quantidade de testes
    - ✓ Recursos humanos / tempo
    - ✓ Quais as plataformas e ferramentas de testes
- **Especificar os testes**
    - ✓ Expectativas
    - ✓ Técnicas a utilizar/não utilizar
    - ✓ Partes da aplicação a considerar
- **Conceber os testes**
    - ✓ Definição dos cenários (casos de testes)
- **Estabelecer condições dos testes (inputs)**

Níveis dos testes de software



Fases de desenvolvimento e testes de software



(CRAIG e JASKIEL, 2002)

Tipo de testes de software

- **Fase de desenvolvimento**
  - ✓ Testes unitários
  - ✓ Testes de integração
- **Fase de produção**
  - ✓ Testes de sistema
- **Fase de certificação**
  - ✓ Testes de aceitação
- **Fase de manutenção**
  - ✓ Testes de regressão

Tipo de testes de software

- **Testes unitários**

  ✓ Tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente. O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenas unidades de código.

- **Testes de integração**

  ✓ Visa provocar falhas associadas às interfaces entre os módulos quando esses são integrados para construir a estrutura do software que foi estabelecida na fase de projeto.

Tipo de testes de software

- **Testes de sistema**

  ✓ Avalia o software em busca de falhas por meio da utilização do mesmo, como se fosse um utilizador final. Dessa forma, os testes são executados nos mesmos ambientes, com as mesmas condições e com os mesmos dados de entrada que um utilizador utilizaria no seu dia-a-dia de manipulação do software. Verifica se o produto satisfaz seus requisitos.

- **Testes de aceitação**

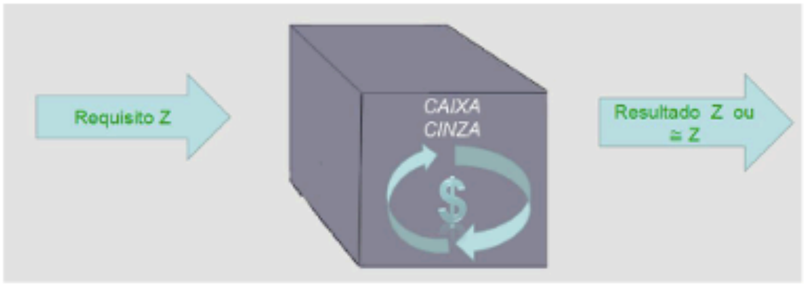
  ✓ São realizados geralmente por um restrito grupo de utilizadores finais do sistema. Estes simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado.

Tipo de testes de software

- **Testes de regressão**

  ✓ Testes de regressão não correspondem a um nível de teste, mas é uma estratégia importante para redução de “efeitos colaterais”. Consiste em aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema. Pode ser aplicado em qualquer nível de teste.

Técnicas de testes de software



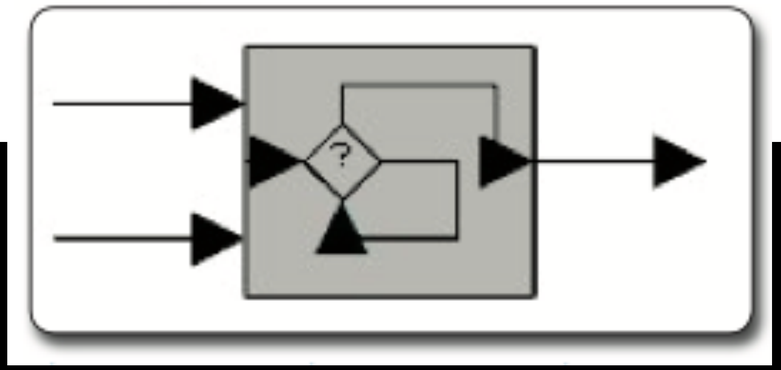
Técnicas de testes de software

- **Estrutural ou caixa branca**

  ✓ Técnica de teste que avalia o comportamento interno do componente de software. Essa técnica trabalha diretamente sobre o código fonte do componente de software para avaliar aspetos tais como: teste de condição, teste de fluxo de dados, teste de ciclos e teste de caminhos lógicos (PRESSMAN, 2005).

  ✓ Os aspectos avaliados nesta técnica de teste dependerão da complexidade e da tecnologia determinada na construção do componente de software, cabendo, portanto, avaliação de outros aspectos além dos citados anteriormente. O engenheiro de testes tem acesso ao código fonte da aplicação e pode construir códigos para efetuar a ligação de bibliotecas e componentes.

  ✓ Este tipo de teste é desenvolvido analisando-se o código fonte e elaborando-se casos de teste que cubram todas as possibilidades do componente de software. Dessa forma, todas as variações originadas por estruturas de condições são testadas.



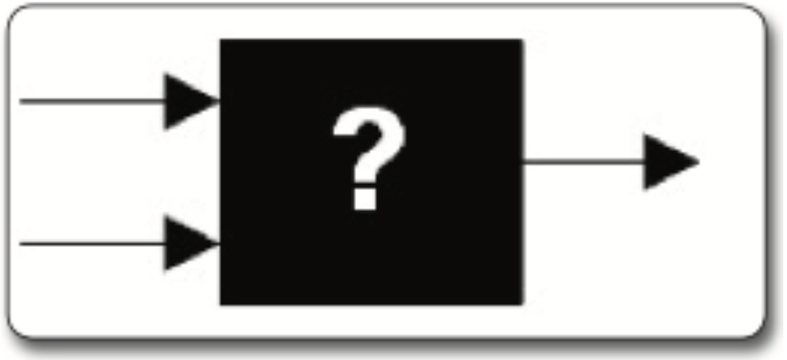
Técnicas de testes de software

- **Funcional ou caixa preta**

  ✓ Técnica de teste em que o componente de software a ser testado é abordado como se fosse uma caixa-preta, ou seja, não se considera o comportamento interno do mesmo. Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido. Haverá sucesso no teste se o resultado obtido for igual ao resultado esperado.

  ✓ O componente de software a ser testado pode ser um método, uma função interna, um programa, um componente, um conjunto de programas e/ou componentes ou mesmo uma funcionalidade. A técnica de teste funcional é aplicável a todos os níveis de teste (PRESSMAN, 2005).

  ✓ Um conjunto de critérios de teste pode ser aplicado aos testes funcionais.



Técnicas de testes de software

- **Caixa cinza**

  ✓ Alguns autores chegam a definir uma técnica de teste caixa cinza, que seria uma mistura de uso das técnicas de caixa preta e caixa branca, mas, como toda execução de trabalho relacionado à atividade de teste utilizará simultaneamente mais de uma técnica de teste, é recomendável que se fixem os conceitos primários de cada técnica.

CI & CD - Continuous integration & Continuous delivery/ deployment

- **Continuous integration (CI)**

  ✓ É uma prática de desenvolvimento de software onde os programadores regularmente fazem o *merge* das suas alterações de código num repositório central/remoto, após o qual compilações e testes automatizados são executados.

  ✓ Os principais objetivos da CI são encontrar e resolver bugs mais rapidamente, melhorar a qualidade do software e reduzir o tempo necessário para validar e lançar novas atualizações de software.

  ✓ A integração contínua concentra-se em *commits* menores e mudanças de código menores para integração.

  ✓ Um programador confirma o código em intervalos regulares, no mínimo uma vez por dia e extrai o código do repositório de código para garantir que seja feito o *merge* no código do *host* local antes de enviar para o servidor de compilação.

  ✓ O servidor de construção executa os vários testes e aceita ou rejeita o *commit* do código.

  ✓ Os desafios básicos da implementação de CI incluem *commits* mais frequentes na base de código comum, mantendo um único repositório de código-fonte, automatizando compilações e automatizando testes.

  ✓ Desafios adicionais incluem testes em ambientes semelhantes aos de produção, fornecendo visibilidade do processo para a equipa e permitindo que os programadores obtenham facilmente qualquer versão do software.

CI & CD - Continuous integration & Continuous delivery/ deployment

- **Continuous deployment/Deployment (CD)**

  ✓ É uma prática de desenvolvimento de software em que as alterações de código são criadas, testadas e preparadas automaticamente para lançamento para produção do produto.

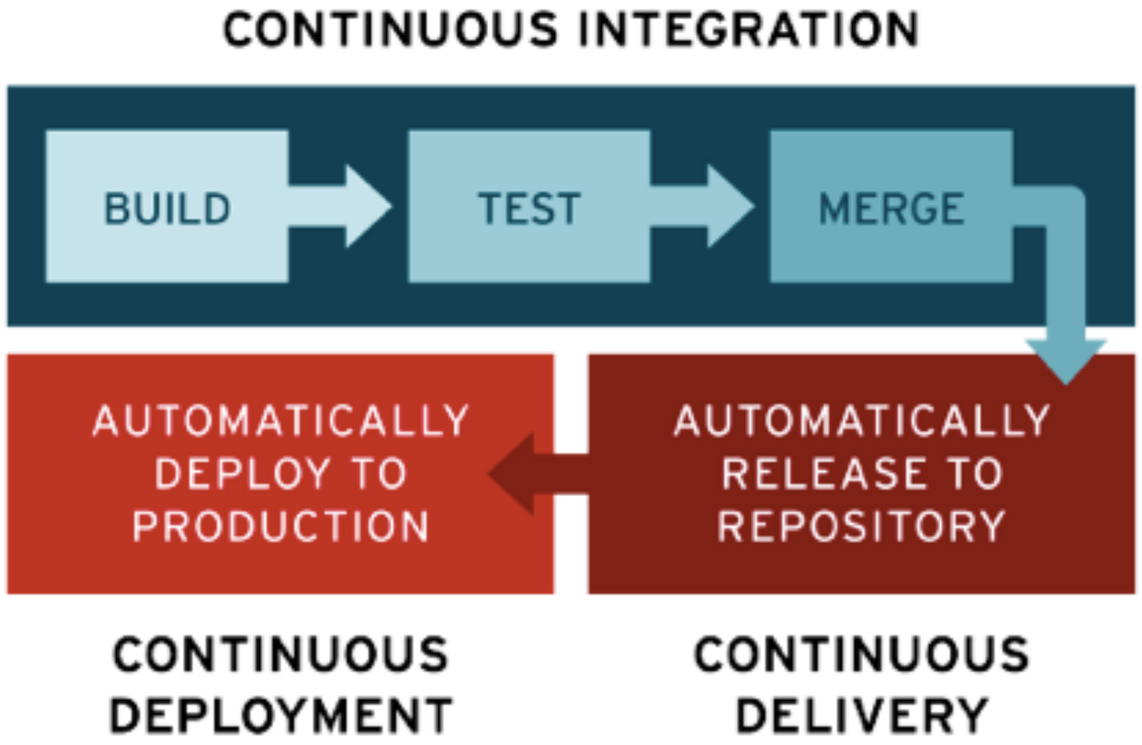
  ✓ Desta forma ***continuous integration*** permite a implementação de todas as alterações de código num ambiente de testes, num ambiente de produção ou ambos após a conclusão do desenvolvimento.

  ✓ ***Continuous deployment*** pode ser totalmente automatizada com um processo de fluxo de trabalho ou parcialmente automatizada com etapas manuais em pontos críticos.

  ✓ Quando ***continuous delivery*** é implementada adequadamente, os programadores têm um artefacto de construção pronto para a implementação que passou por um processo de teste padronizado.

  ✓ Com c***ontinuous deploymente***, as revisões são implementadas num ambiente de produção automaticamente, sem a aprovação explícita de um programador, tornando todo o processo de lançamento de software automatizado. Isso, por sua vez, permite um ciclo contínuo de *feedback* do cliente logo desde o início do ciclo de vida do produto.

CI & CD - Continuous integration & Continuous delivery/ deployment



Benefícios de Continuous delivery

- **Automatiza o processo de lançamento de software**

  ✓ O CD fornece um método para a equipa fazer o check-in do código que é criado, testado e preparado automaticamente para produção, de modo que a entrega do software seja eficiente, resiliente, rápida e segura.

- **Melhora a produtividade do programador**

  ✓ As práticas do CD ajudam a produtividade da sua equipe, libertando os programadores de tarefas manuais, permitindo o foco para o fornecimento de novos recursos no software.

Benefícios de Continuous delivery

- **Melhora a qualidade do código**

  ✓ O CD pode ajudar a descobrir e resolver bugs no início do processo de entrega, antes que se transformem em problemas maiores posteriormente.

  ✓ A equipa pode facilmente executar tipos adicionais de testes de código porque todo o processo foi automatizado. Com a disciplina de mais testes com mais frequência, as equipes podem iterar mais rapidamente com *feedback* imediato sobre o impacto das mudanças.

  ✓ Isso permite que as equipes gerem código de qualidade com alta garantia de estabilidade e segurança.

  ✓ Os programadores saberão, por meio de *feedback* imediato, se o novo código funciona e se quaisquer alterações importantes ou *bugs* foram introduzidos.

  ✓ Erros detectados no início do processo de desenvolvimento são os mais fáceis de corrigir.

Benefícios de Continuous delivery

- **Entregar atualizações mais rápido**

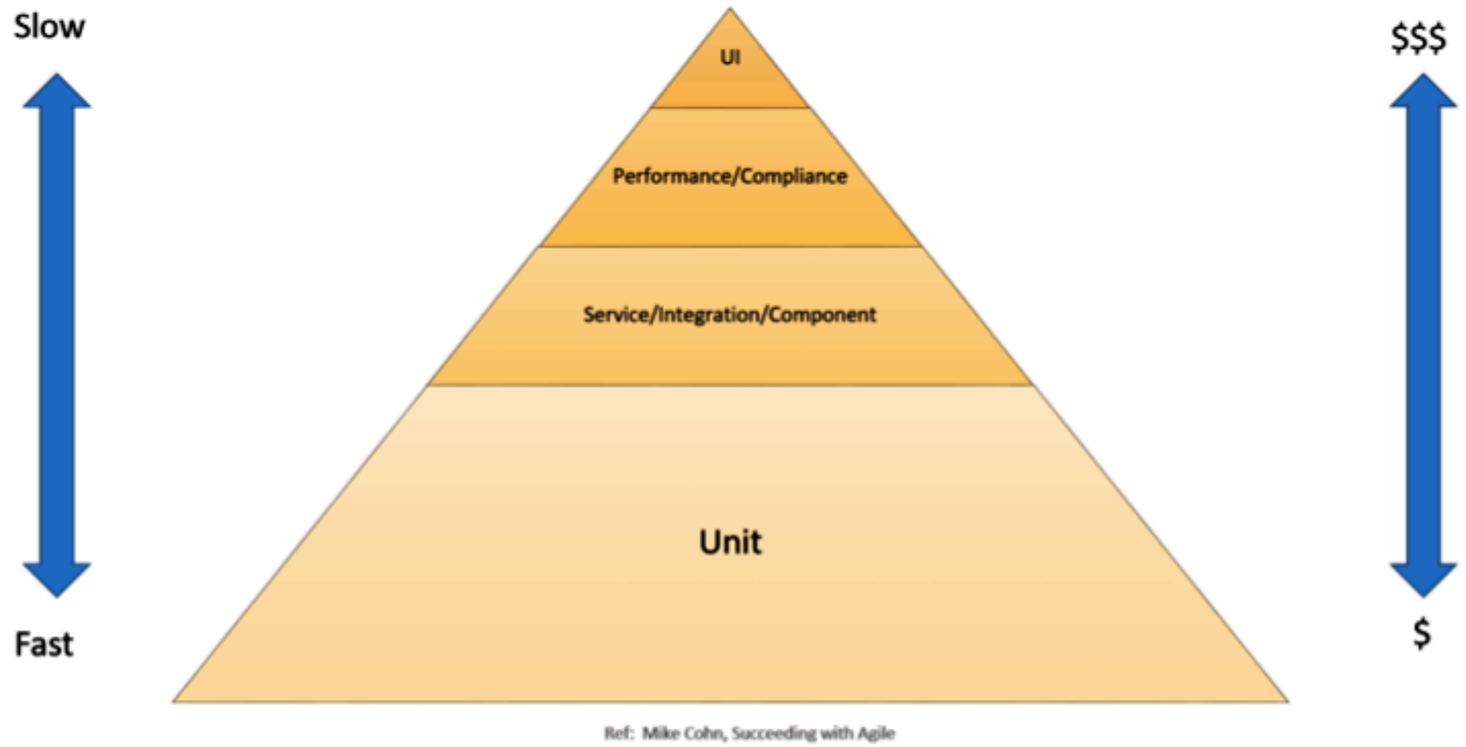
  ✓ O CD ajuda a equipa a fornecer atualizações aos clientes com maior rapidez e frequência.

  ✓ Quando o CI / CD é implementado, a velocidade de toda a equipa, incluindo o lançamento de recursos e correções de *bugs*, aumenta.

  ✓ As empresas podem responder mais rapidamente às mudanças do mercado, desafios de segurança, necessidades do cliente e pressões de custo.

  ✓ Por exemplo, se um novo recurso de segurança for necessário, a equipa pode implementar CI / CD com testes automatizados para apresentar a correção de forma rápida e confiável aos sistemas de produção com alta confiabilidade. O que costumava levar semanas e meses, agora pode ser feito em dias ou mesmo horas.

Benefícios de Continuous delivery



Refactoring

- ...
- ...

Engenharia de Software

Testes de software

Marta Martinho 2020