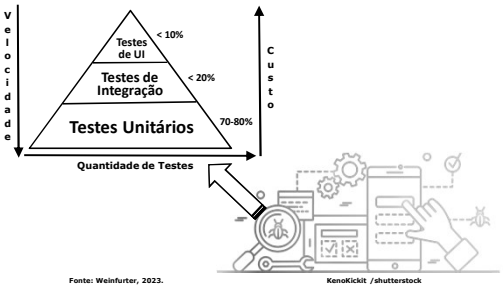
Aula 4

Teste de Software

Profª Maristela Weinfurter

Conversa Inicial



Fonte: Weinfurter, 2023.

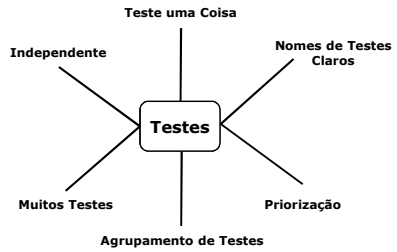
KenoKickit /shutterstock

- Todas as classificações ou categorizações nas quais os vários tipos de testes se encontram fazem parte dessa visão sistêmica
- Isso nos remete à ideia de que o caminho de testes que garantam a qualidade de nosso software deve ser bem planejado e adaptado à realidade de nosso projeto e de nossa empresa

Casos de testes

- Cada teste compreende quatro elementos
  - Pré-requisitos
  - Configurar
  - Procedimento
  - Resultado

CkyBe/shutterstock



Fonte: Weinfurter, 2023.

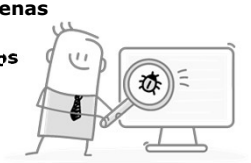
- Independência de outros testes (independente)
  - Idealmente, cada teste passaria ou falharia por conta própria. Dessa forma, onde houver uma falha haverá um indicador claro, em vez de muitos testes falharem de uma só vez

Semanche/shutterstock

- Cada teste deve verificar uma coisa (teste uma coisa)
  - Além de ser independente dos outros, cada teste deve ter um propósito e verificar apenas uma parte da funcionalidade, na medida do possível

Semanche/shutterstock

- Não ter medo de fazer muitos testes (muitos testes)
  - Um corolário para cada teste verificando apenas uma coisa é o que precisamos de muitos deles. Novamente, tudo bem. Disponha-os claramente



Semanche/shutterstock

- Agrupar seus testes (agrupamento)
  - Com muitos testes para gerenciar, precisamos verificar se eles estão divididos em diferentes seções e pastas para que saibamos onde encontrá-los e onde novos testes devem ser adicionados

Semanche/shutterstock

- Fornecer nomes claros aos testes (nomes claros)
  - Ao examinar uma lista de falhas, devemos facilitar a identificação do que deu errado ao incluirmos nomes claros. Isso pode significar que precisamos de nomes longos e tudo bem

Semanche/shutterstock

- Priorizar os casos de teste (priorização)
  - Planejar para termos muitos casos de teste, mas não que todos sejam iguais. Se eles começarem a demorar muito para serem executados, concentre-se nos testes de caminho feliz e nas áreas de fraqueza conhecidas onde encontramos *bugs* anteriormente

Semanche/shutterstock

13/34

Preparação do ambiente

14/34

- A preparação do ambiente de teste de software é uma fase do processo de testes e garante o funcionamento correto do software a ser colocado em produção
- Podemos testar nas máquinas do desenvolvedor, onde o novo código foi escrito segundos antes, em sua instância de execução ao vivo disponível para seus usuários ou em vários locais intermediários, como áreas de teste dedicadas

15/34

- Exploraremos como atingir objetivos conflitantes simultaneamente:
  - Entendendo os diferentes níveis de teste
  - Definindo casos de teste
  - Etapas do teste prescritivo e descritivo
  - Avaliando diferentes ambientes de teste
  - Definindo versão, configuração e ambiente corretos
  - Realizando testes sistemáticos
  - Testes no ciclo de lançamento
  - Usando curiosidade e feedback

16/34

- Vantagens e desvantagens dos ambientes de teste temporário

VANTAGENS	DESVANTAGENS
▪ Garantido para estar executando o código mais recente ▪ Sempre comece a partir de um estado conhecido ▪ Sem manutenção contínua ▪ Sem custos de funcionamento contínuos ▪ Fácil de testar várias alterações em paralelo	▪ Todos os usuários precisam ser capazes de criar o ambiente ▪ Precisa de configuração para testes fora do padrão ▪ Nenhum uso histórico real

Fonte: Weinfurter, 2023.

17/34

- Vantagens e desvantagens dos ambientes de teste permanente

VANTAGENS	DESVANTAGENS
▪ Rápido de usar para testes curtos ▪ Mais fácil de monitorar ▪ Pode encontrar problemas devido à execução de longo prazo ▪ Acumular dados históricos	▪ Problemas afetam muitos usuários simultaneamente ▪ Requer manutenção ▪ Pode conter dados incorretos de compilações de desenvolvimento ▪ Exigir atualizações

Fonte: Weinfurter, 2023.

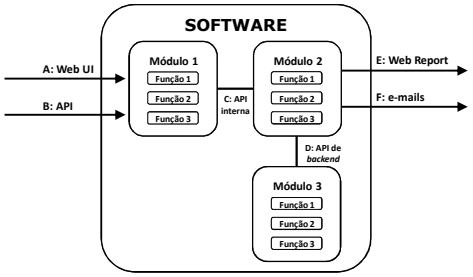
18/34

- Uma das maiores perdas de tempo ao testar é usar a versão ou configuração errada
- O teste pode ocorrer rapidamente quando tudo está funcionando conforme o planejado
- Assim que algo der errado, paramos para investigar o que deve ser reservado para *bugs* genuínos. Qualquer coisa que possamos consertar sozinhos devemos fazê-lo primeiro

Semanche/shutterstock

Teste de integração

- Componentes e módulos podem muito bem funcionar corretamente de forma isolada, mas, ao serem integrados, podem gerar problemas ou defeitos inesperados. Logo, nossos testes de integração permitem a identificação e correção dessesproblemas antes que entreguemos o produto final ao nosso cliente



Fonte: Weinfurter, 2023.

- O teste de integração isola um módulo específico
- Para testar o Módulo 1, precisamos controlar suas três interfaces, A, B e C. A interface da web e a API são voltadas para o público e provavelmente serão bem documentadas e compreendidas

VANTAGENS	DESVANTAGENS
▪ Pode desacoplar seções de grandes projetos ▪ Combinar testes com unidades funcionais ▪ Teste de correspondência com unidades organizacionais ▪ As falhas são isoladas ▪ Mais simples que os testes de sistema ▪ Agnóstico em relação à implementação ▪ Não requer um sistema completo	▪ Leva tempo para configurar *stubs* e *mocks* ▪ Risco de ser irrealista ▪ Toma tempo do desenvolvedor ▪ Mais complicado que testes unitários ▪ Cobertura limitada

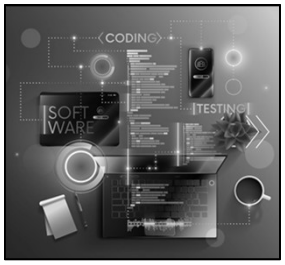
Fonte: Weinfurter, 2023.

- Os testes de integração são mais realistas do que os testes unitários porque testam um módulo ou serviço inteiro
- Com muitas funções trabalhando juntas, podemos testar suas interações sem exigir que todo o sistema esteja em execução

Semanche/shutterstock

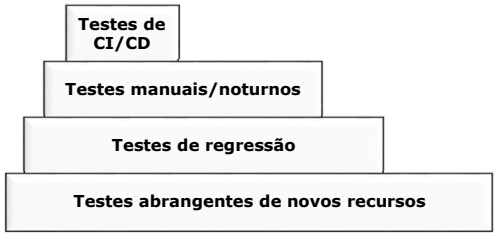
Teste de regressão

- Os testes de regressão incluem apenas os testes que desejamos executar quando um recurso for modificado



Andrey Suslov/shutterstock

- Depois dos testes de regressão, temos os testes manuais/noturnos. Eles têm um limite de tempo – por exemplo, 12 horas para uma execução noturna –, portanto, precisam ser priorizados com cuidado
- Eles encontram problemas que podem afetar usuários ativos e seriam pequenas interrupções, mas abrangem casos que consomem muito tempo para serem incluídos nos testes de CI/CD



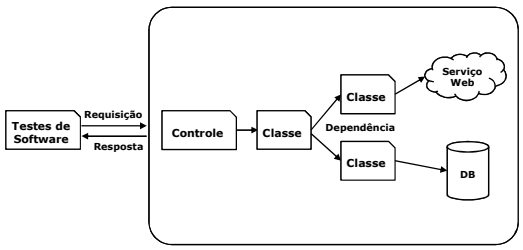
Fonte: Weinfurter, 2023.

- Na prática, o teste de regressão completo que executa todos os testes existentes geralmente é muito caro e leva muito tempo, especialmente quando, como já como mencionado, estão envolvidos testes manuais
- Portanto, estamos procurando critérios que nos ajudem a decidir quais casos de teste herdados podem ser ignorados sem perder muita informação

Semanche/shutterstock

Teste de sistema

- Após a conclusão do teste de integração, o próximo nível de teste é o teste do sistema. Esse nível verifica se o sistema completo e integrado realmente atende aos requisitos especificados
- Os testes de sistema fazem parte do topo da pirâmide de testes e são responsáveis por testes que simulam o uso de um sistema por um usuário/*stakeholder* real. São muito conhecidos como *testes ponta-a-ponta* (*end-to-end*) ou até como *testes de interfaces*. São testes dispendiosos que demandam muito esforço para a implementação e mais tempo para a execução



Fonte: Weinfurter, 2023.

- A visão sistêmica da abordagem de testes, validações e verificações de software nos proporciona exatamente a ideia de início, meio e fim do processo como um todo

Semanche/shutterstock