

# Orientações técnicas Vídeo

- **Orientações técnicas Vídeo:** OBS Studio.
- **Resolução:** 1280x720 • Ideal: 1920x1080.
- **Formato de gravação:** AVI ou MP4.
- **Qualidade de gravação:** O padrão do OBS Studio é “A mesma da transmissão”, mude para “Qualidade alta, arquivo normal”.
- Aumentar o tamanho da fonte de terminal/IDE.
- **Microfone:** Teste isolamento acústico (algum ruído é tolerável) e volume da voz entre 5-15 dB, se possível.

# Fundamentos de Qualidade de Software

Carolina Santana Louzada

Engenheira de Qualidade de Software - UOLEdtech

# Mais sobre mim

- Graduada em Engenharia de Computação- UFS
- Fazendo especialização em qualidade e desenvolvimento de software
- Qualidade de software -> automação
- Educação + tecnologia
- Jogos + música + aprender novas atividades
- LinkedIn -> [Carolina Santana Louzada | LinkedIn](#)

# Objetivo do curso

Compreender fundamentos e normas fundamentais da área de qualidade, assim como aprofundar atividades de um analista ou engenheiro de qualidade software no mercado de trabalho . Introduzir níveis e tipos de teste e como estes se inserem no contexto da garantia qualidade.

# Pré-requisitos

- Dedicação e vontade de aprender
- Um pouquinho de paciência
- Mente aberta

# Percurso

## Aula 1

O que é qualidade de software?

## Aula 2

Gerenciamento de defeitos

## Aula 3

Introdução aos testes de software

# Dúvidas durante o curso?

- > Fórum do curso
- > Comunidade online (Discord)



## Aula 1

# O que é qualidade de software?

// Fundamentos de Qualidade de Software



# Objetivos

- Definindo qualidade
- Normas e padrões de qualidade
- Medindo a qualidade
- Processos de gerenciamento de qualidade de software

Aula 1 . Etapa 1

# Definindo qualidade

// Fundamentos de qualidade de software

# Definições na literatura

- ★ [NBR/ISO 9000:2005](#): grau no qual um conjunto de características inerentes satisfaz a requisitos

# Definições na literatura

- ★ Peters(2002) : “A qualidade de software é avaliada em termos de atributos de alto nível chamado fatores, que são medidos em relação a atributos de baixo nível, chamados critérios.”
- ★ Sanders(1994): “ Um produto de software apresenta qualidade dependendo do grau de satisfação das necessidades dos clientes sob todos os aspectos do produto.”

# Definições na literatura

- ★ Pressman: “Qualidade de software é a conformidade a requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais.”

# Definições na literatura

- ★ [ISO/IEC 25010:2011](#) : “capacidade do produto de software de satisfazer necessidades declaradas e implícitas sob condições especificadas”
- ★ [IEEE Standard\(2014\)](#): “ o grau em que um produto de software atende aos requisitos estabelecidos; no entanto a qualidade depende do grau em que esses requisitos representam com precisão as necessidades, desejos e expectativas das partes interessadas ”

# Definições na literatura

- ★ Aspectos importantes:
  - requisitos de software são a base para medir qualidade
  - padrões especificados definem conjunto de critérios de desenvolvimento
  - existem requisitos implícitos que não são mencionados que afetam diretamente a qualidade

# Percepções de qualidade

<b>Visão transcendental</b>	Qualidade é reconhecida através de experiência, mas sem uma definição ou metrificação.
<b>Visão do usuário</b>	É personalizado de acordo com a necessidade do usuário.
<b>Visão de manufatura</b>	Qualidade é relacionada com conformidade aos requerimentos
<b>Visão de produto</b>	Produto com boas propriedades internas metrificáveis terá boas qualidade externas.
<b>Visão baseada em valor</b>	Representa o 'custo-benefício' na visão do cliente.



Aula 1 . Etapa 2

# As normas e padrões de qualidade

// Fundamentos de qualidade de software

# O que são normas técnicas?

Documentos publicados por organizações profissionais que objetivam padronizar determinadas atividades, processos, produtos, etc...

**A chave é a  
comunicação**



# Instituições importantes

→ IEEE: *“Institute of Electrical and Electronics Engineers”*



→ ISO: *“International Organization for Standardization”*



→ IEC: *“International Electrotechnical Commission”*



# As normas para qualidade de software

Família ISO 9000	
ISO 9000	Descreve fundamento de sistemas de gestão de qualidade e suas terminologias
ISO 9001	Especifica requisitos para sistema de gestão de qualidade
ISO 9004	Diretrizes que consideram eficácia e eficiência do sistema de gestão da qualidade.
ISO 9126	Modelo de qualidade de produto de software

# As normas para qualidade de software

ISO/IEC 14598	Processo de avaliação de produtos de software na visão do desenvolvedor, adquirente e avaliador
ISO/IEC/IEEE 12207:2017 ISO/IEC/IEEE 15288:2015	Processos de ciclo de vida do software
ISO 19011	Diretrizes sobre auditoria de sistemas de gestão da qualidade de software
IEEE 1012:2016	Verificação e validação para sistemas, software e hardware

# As normas para qualidade de software

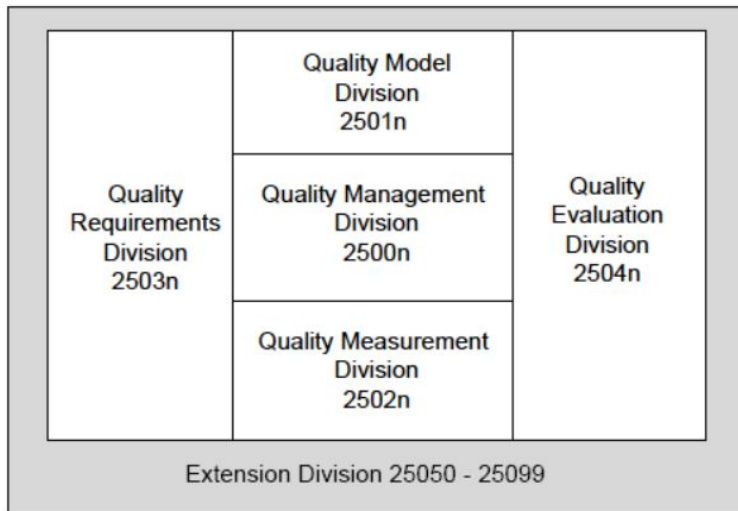
IEEE 730:2014	Requerimentos para planejamento, controle e execução de processos de garantia de qualidade de software
ISO/IEC/IEEE 15289:2019	Foco no processo de gerenciamento de informação
ISO/IEC/IEEE 29119:2013	Vocabulário, processo, documentação, modelos e técnicas para teste

# As normas para qualidade de software

Família SQuaRE: ISO/IEC 25000-25099  
(*System and Software Quality Requirements and Evaluation*)

Substitui ISO/IEC 9126

- Requerimentos de qualidade
- Modelo de Qualidade
- Gerenciamento de qualidade
- Metrificação de qualidade
- Avaliação de qualidade



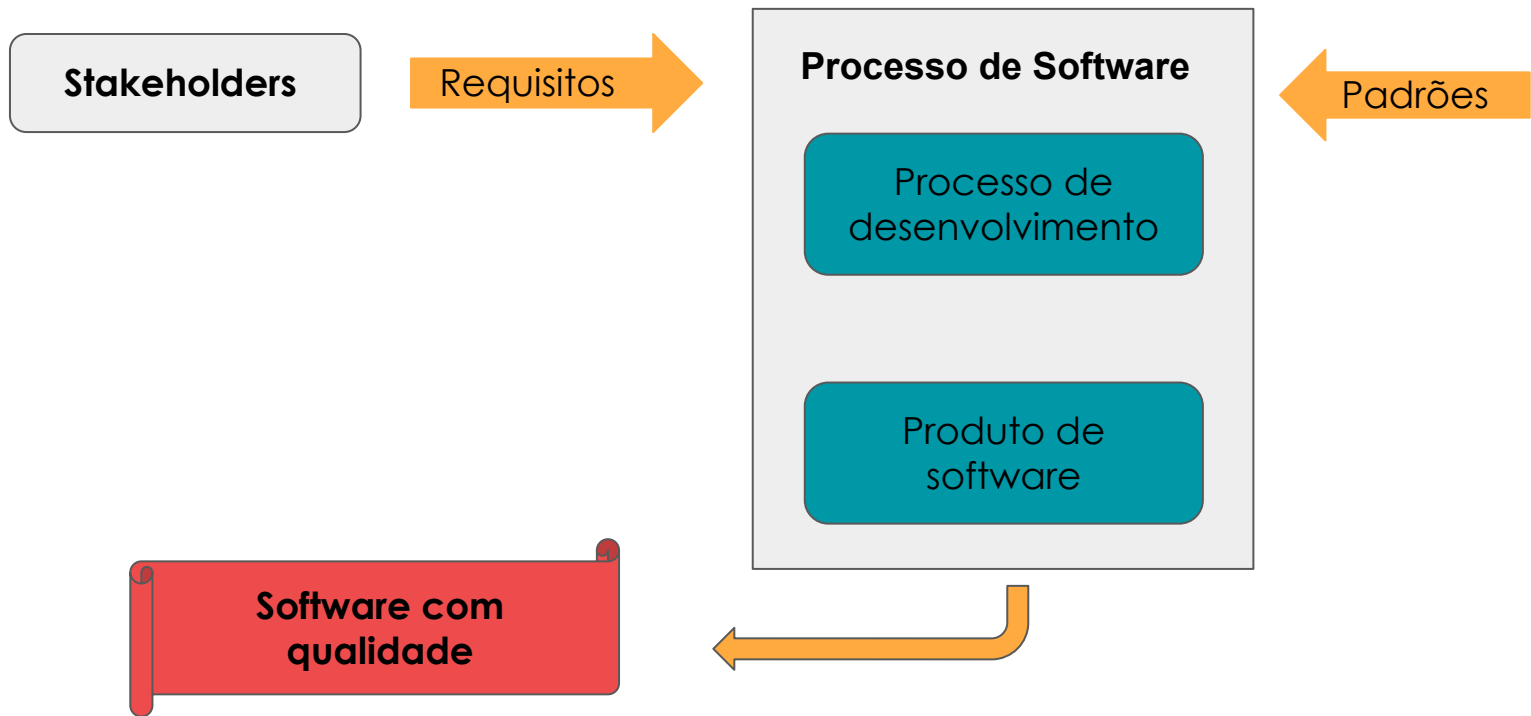
Aula 1 . Etapa 3

# Medindo a qualidade

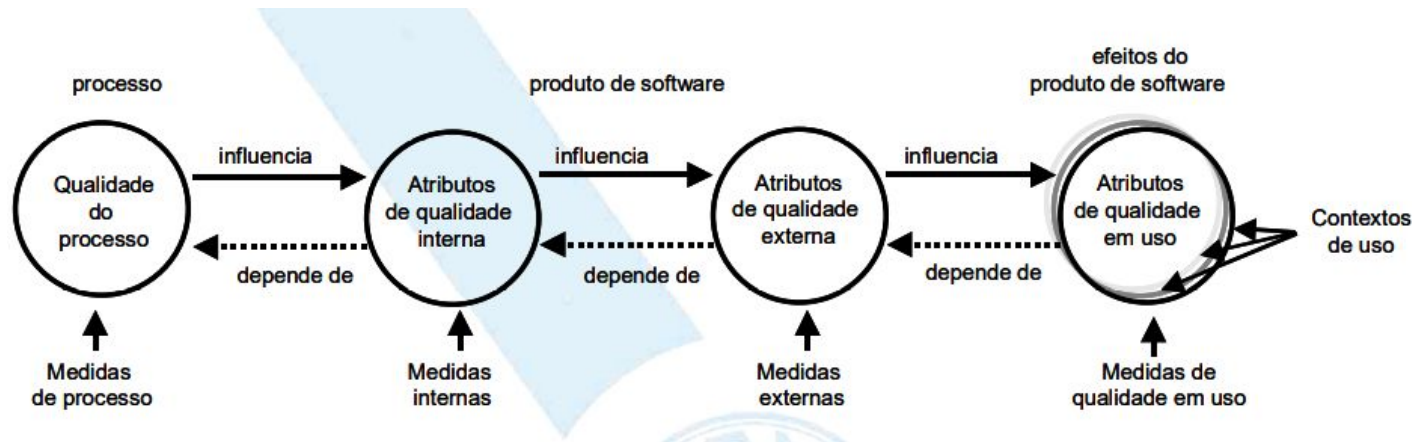
// Fundamentos de qualidade de software



# Processo da qualidade

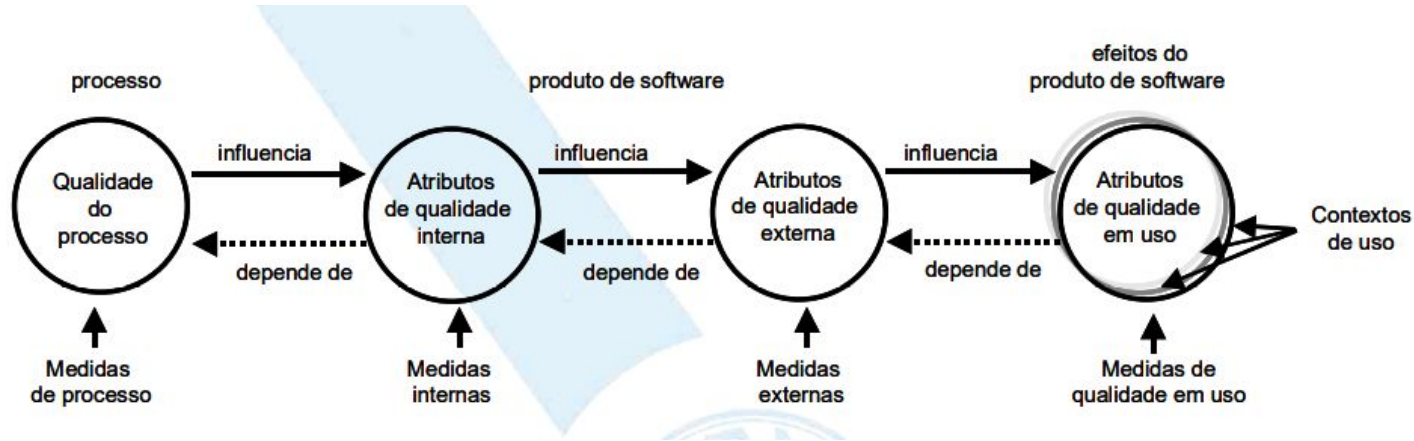


# Medindo qualidade



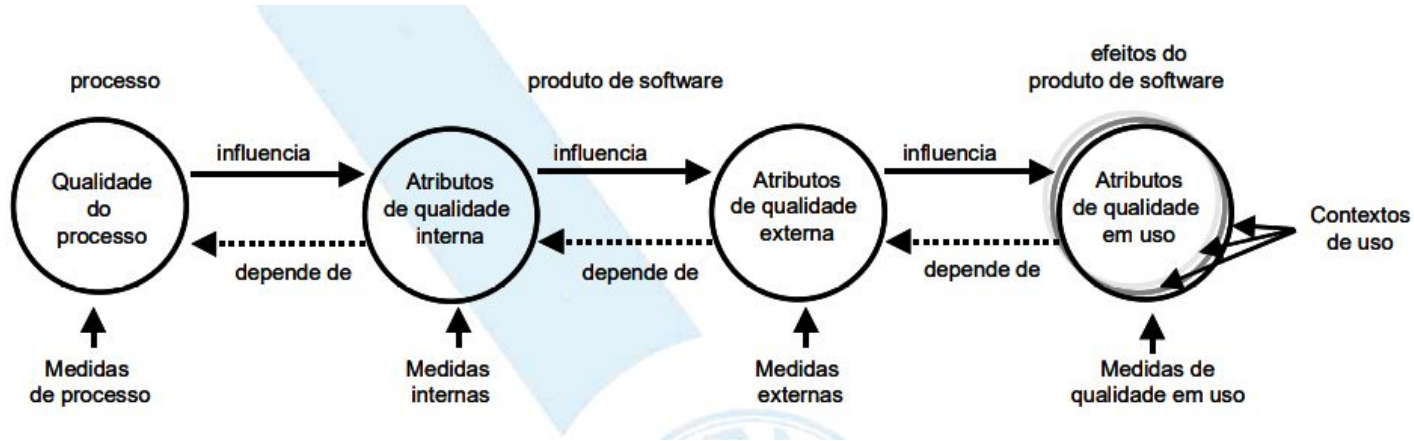
- Qualidade interna: totalidade das características do software do ponto de vista interno
- Métricas internas: podem ser aplicadas a um produto de software não executável, como especificações e código-fonte. Servem para avaliar a qualidade do produto antes do produto se tornar executável. São também indicadores de atributos externos.

# Medindo qualidade



- Qualidade externa: totalidade das características do produto do ponto de vista externo, incluindo requisitos derivados das necessidades do usuário e dos requisitos de qualidade em uso
- Métricas externas: utilizam medidas de um produto de software derivadas de medidas do comportamento do sistema através de testes, operação e observação do produto

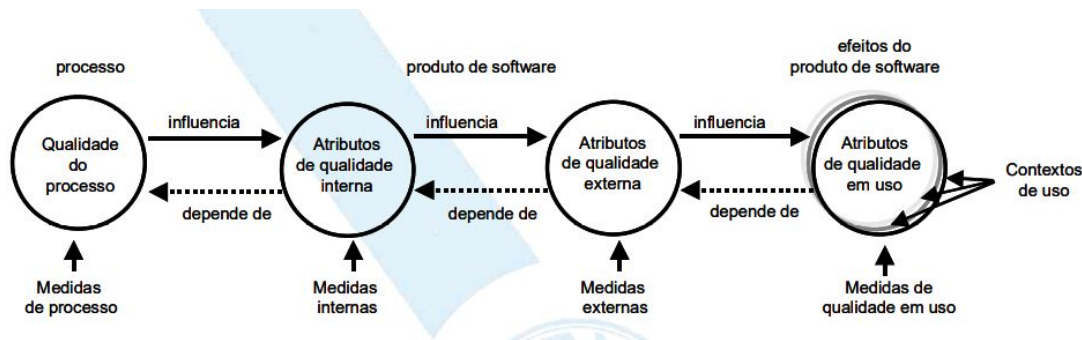
# Medindo qualidade



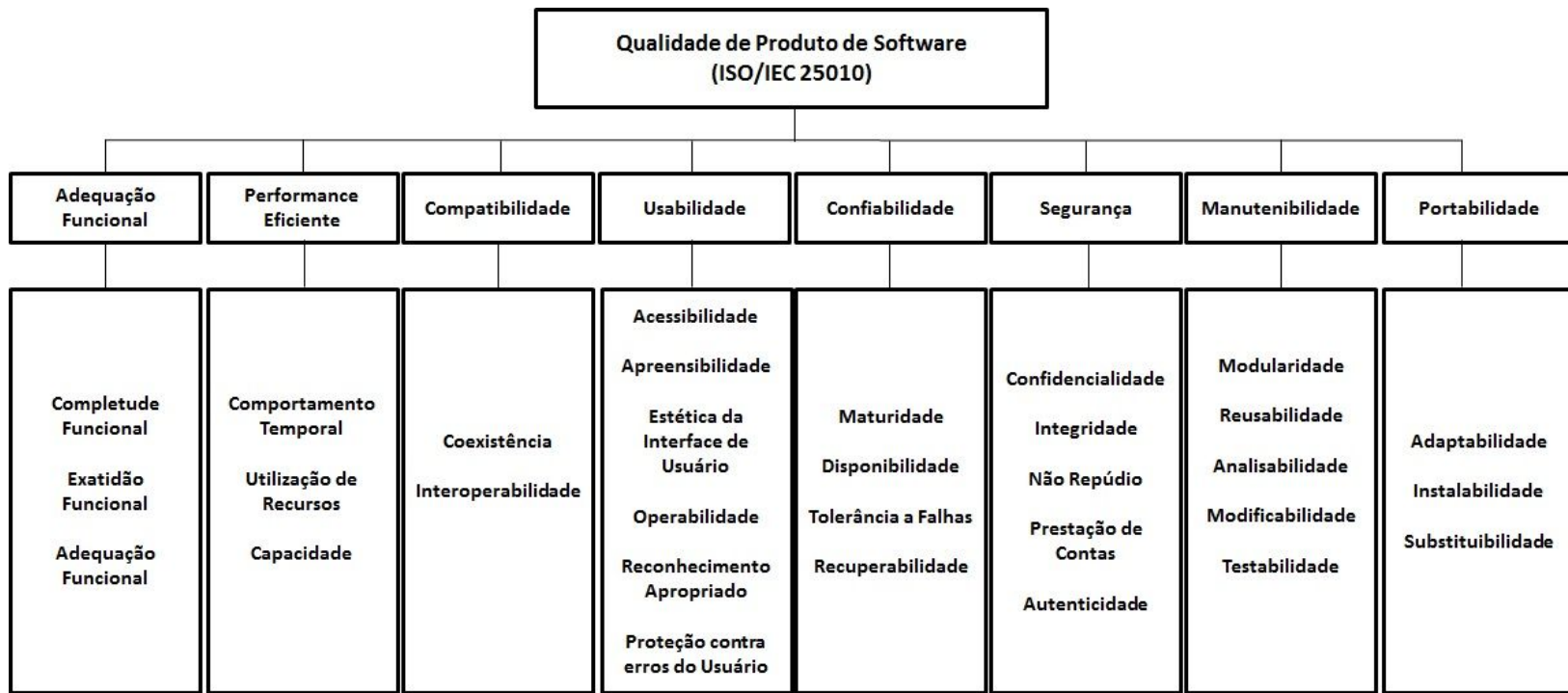
- Qualidade em uso: visão da qualidade do ponto de vista do usuário, em um ambiente e contexto de uso específicos
- Métricas de qualidade em uso: medem o quanto um produto atende às necessidades de usuário especificados

# Olhando para ISO/IEC 25010

- Modelo de qualidade de produto de software: composto por 8 características subdivididas em sub-características
- Modelo de qualidade em uso: composto de 5 características e suas subcaracterísticas



# ISO/IEC 25010 - Qualidade do produto



# ISO/IEC 25010 - Qualidade em uso



## Aula 1 . Etapa 4

# Processos de gerenciamento de qualidade de software

// Fundamentos de qualidade de software



# Gerenciamento de qualidade de software

- ★ Conjunto de todos os processos que garantem que os produtos, serviços e o ciclo de vida vão de encontro com os objetivos da qualidade e forma a alcançar a **satisfação do usuário**.
- ★ Atividades do gerenciamento:
  - Planejamento de qualidade
  - Garantia de qualidade
  - Controle de qualidade
  - Melhoria de processos

# Planejamento

Determinar:

- padrões e processos de qualidades a serem utilizados
- metas específicas de qualidade
- esforço e organização de atividades

# Garantia de qualidade

- Atividades de que definem e avaliam a adequação dos processo de software de forma a prover evidências que estabelecem confiança no produto produzido.

# Controle de qualidade

- Examinação de artefatos do projeto para determinar se os padrões acordados estão sendo seguidos
- Avaliação de produtos intermediários e do produto final

# Melhorias de processos

- Se preocupa com melhorias na eficiência, efetividade e quaisquer características que tenham como meta principal a melhoria da qualidade de software

# Outra perspectiva

1. Aplicação de métodos técnicos
2. Realização de revisões técnicas formais
3. Atividades de testes de software
4. Aplicação de padrões
5. Controle de mudanças
6. Medição
7. Manutenção de registros e relatórios

## Aula 2

# Gerenciamento de defeitos

// Fundamentos de Qualidade de Software

# Objetivos

- Falando em controle de qualidade
- Caracterizando defeitos
- Ciclo de vida do bug
- Ferramentas de suporte



## Aula 2 . Etapa 1

# Falando em controle de qualidade

// Fundamentos de qualidade de software

# Controle de qualidade

- Análise estática: avaliação de documentação do software e código-fonte -> métodos formais
- Análise dinâmica: relacionado a técnicas com o código em execução

Validação X Verificação

**Verificação:** Garantir que o produto está sendo construído corretamente

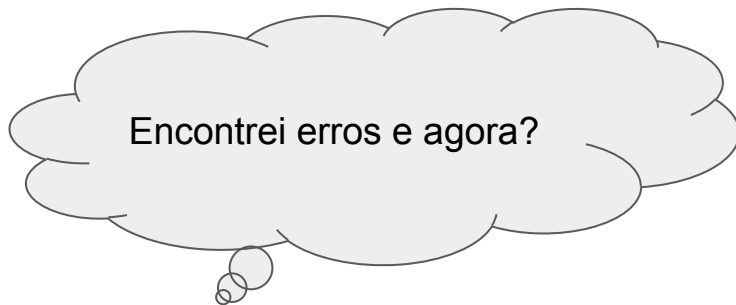
**Validação:** O produto correto está sendo construído.

# Controle de qualidade

- Análise estática: avaliação de documentação do software e código-fonte -> métodos formais
  - ◆ Code review
  - ◆ Ferramentas de automação de processos de verificação de código
  - ◆ Análise de histórias e modelagens

# Controle de qualidade

- Análise dinâmica: relacionado a técnicas com o código em execução
  - ◆ Finalmente nossos queridos testes!



## Aula 2 . Etapa 2

# Caracterizando defeitos

// Fundamentos de qualidade de software

# Rastreamento de defeitos

- Entendimento do produto e dos tipos de defeitos encontrados
- Facilitar correção do processo ou do produto
- Reportar status do produto
- Alinhamento de revisões pelo time de desenvolvimento

# O que é defeito?

- Genericamente significa qualquer tipo de anomalia encontrada no produto
- Outras definições:
  - ◆ Erro: Ação humana que produz um resultado incorreto
  - ◆ Defeito: Imperfeição ou deficiência relacionada aos requerimentos e especificações do produto que se
  - ◆ Falha de sistema: Evento no qual o sistema não executa uma função sob limites específicos

Importância de padronizar  
definições na equipe

# Motivos para erros

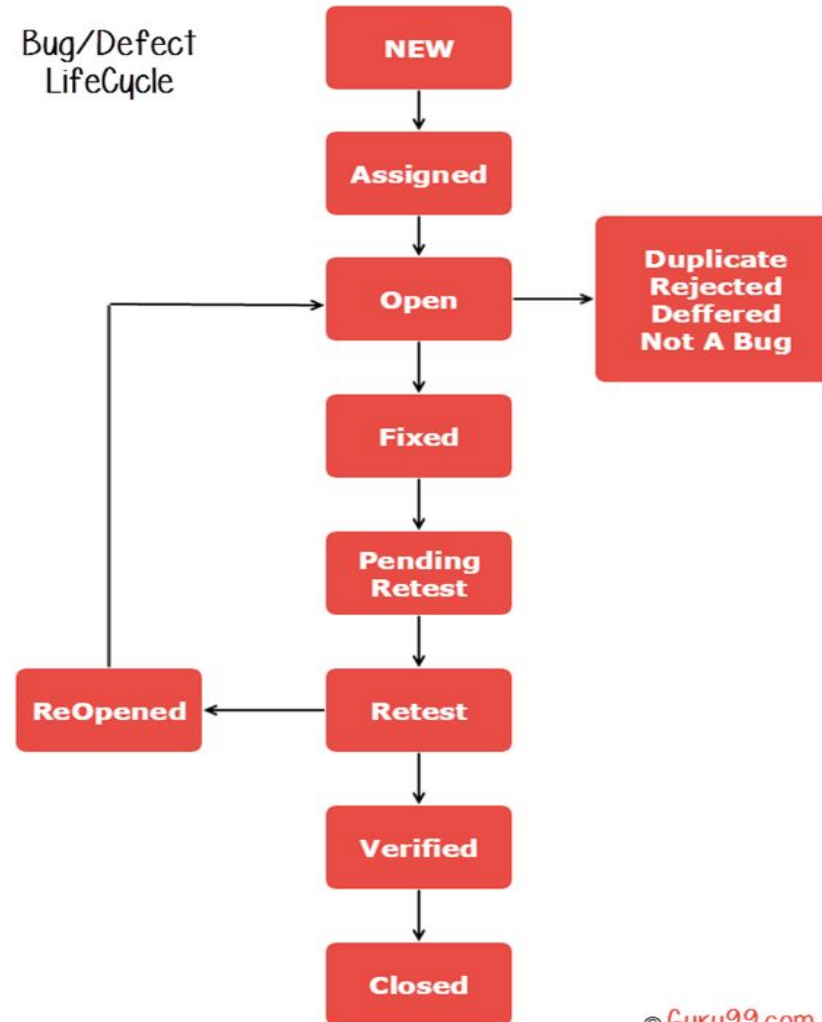
- Pressão do tempo
- Falha humana
- Inexperiência e/ou falta de qualificação
- Falta de comunicação
- Complexidade de código, modelagem, arquitetura...
- Complexidade de tecnologia
- Condições ambientes inesperadas



## Aula 2 . Etapa 3

# Ciclo de vida do bug: do rastreio ao reporte

// Fundamentos de qualidade de software

Bug/Defect  
LifeCycle

# Ciclo de vida

1. New: Defeito é identificado e cadastrado pela primeira vez
2. Assigned: defeito é atribuído para desenvolvedor avaliar
3. Open: desenvolvedor inicia análise e correção
4. Fixed: Desenvolvedor finaliza correção
5. Pending Retest: Estado de espera para o time de teste
6. Retest: Estado de execução do reteste
7. Verified: Defeito corrigido
8. Reopen: Defeito não-corrigido.
9. Closed: Corrigido + testado + aprovado
10. Duplicate: efeito já encontrado anteriormente
11. Rejected: Defeito não é novo.
12. Deferred: Será corrigido em versões futuras.
13. Not a bug: Quando a anomalia não é de fato um erro depois de analisado

# Considerações importantes

- ★ Os processos se adequam ao que o seu time e seu produto precisam!
- ★ O time precisa estar de acordo e entender todo o fluxo de rastreamento de defeitos
- ★ Os defeitos podem e devem ser rastreados em qualquer momento do ciclo de vida do processo de software.
- ★ Principais objetivos dos reports de defeitos:
  - Provê às partes interessadas informações a respeito do evento anômalo de forma a tentar isolar, reproduzir e corrigir o problema ou o potencial problema.
  - Provê meios para rastrear a qualidade do produto e o impacto destes na atividades de testes e retestes
  - Provê ideias para melhoria no processo de desenvolvimento e testes

# Considerações importantes

- ★ Boa comunicação é essencial!
- ★ Uso eficiente de ferramenta de rastreo e report de bugs
- ★ Comprometimento proativo da equipe no gerenciamento dos defeitos



# Informações de um reporte de defeito

- ★ Um identificador único
- ★ Título resumindo o problema
- ★ Data/autor
- ★ Identificação do item sob teste e do ambiente
- ★ Fase do ciclo de vida no qual o defeito foi observado
- ★ Descrição completa do defeito para reprodução
- ★ Evidências de auxílio na resolução:
  - logs
  - dumps de banco de dados
  - screenshots
  - gravações
- ★ Resultado esperado
- ★ Severidade
- ★ Urgência/Prioridade
- ★ Estado do defeito
- ★ Conclusões/Sugestões
- ★ Impactos
- ★ Histórico
- ★ Referência do teste

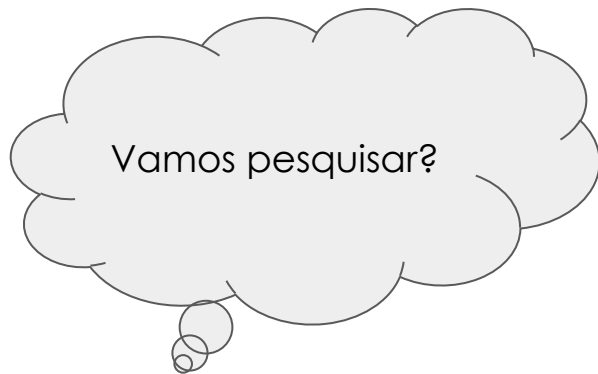
## Aula 2 . Etapa 4

# Ferramentas de suporte

// Fundamentos de qualidade de software

# Algumas ferramentas úteis

- [Bugzilla](#) - gratuito
- [Jira](#) - gratuito e pago
- [Trac](#) - gratuito
- [Redmine](#) - gratuito
- [Asana](#) - gratuito e pago
- [Trello](#) - gratuito e pago
- [Backlog](#) - gratuito e pago
- [ReQtest](#) - pago
- [Mantis](#) - gratuito
- [Axosoft](#) - pago
- [Etraxis](#) - gratuito
- [Lighthouse](#) - pago
- [Azure Devops](#) - pago





## Aula 3

# Introdução aos testes de software

// Fundamentos de Qualidade de Software

# Objetivos

- Conceitos e objetivos
- Processo de teste
- Níveis de teste
- Tipos de teste
- Técnicas de teste

## Aula 3 . Etapa 1

# Teste: conceitos e objetivos

// Fundamentos de qualidade de software

# O que é teste?

- Processo de avaliar e reduzir risco de falhas de software em operação
- Faz parte do controle de qualidade
- O processo de teste não diz respeito somente ao ato de executar um teste



# Objetivos gerais

- Evitar defeitos e avaliar produtos de trabalho
- Verificar cumprimento de requisitos
- Validar se produto funciona como cliente espera
- Criar confiança no nível de qualidade do objeto testado
- Redução de riscos
- Atuar junto ao cliente para tomada de decisões

# Teste X depuração

- A execução de testes pode mostrar falhas causadas por defeitos de software
- Depuração já é um processo de investigação e correção do erro no processo do desenvolvimento do código
- Essas atividades variam de acordo com a metodologia utilizada na equipe

# Princípios de teste

1. Teste mostra presença de defeitos e não a ausência
2. Testes exaustivos são impossíveis
3. Testes iniciais economizam tempo e dinheiro
4. Defeitos se agrupam
5. O mesmo teste não encontra novos defeitos -> atenção com testes de regressão
6. O teste depende do contexto
7. Ausência de erros é ilusão

## Aula 3 . Etapa 2

# O processo de teste

// Fundamentos de qualidade de software



# Fatores de influência

- ★ Modelo de ciclo de vida
- ★ Níveis e tipos de teste
- ★ Risco de produto e projeto
- ★ Domínio do negócio
- ★ Restrições operacionais
- ★ Políticas e práticas organizacionais
- ★ Normas internas e externas

# Atividades de teste

1. Planejamento
2. Monitoramento e controle do teste
3. Análise
4. Modelagem
5. Implementação
6. Execução
7. Conclusão

# Planejamento do teste

- Definir propósitos do teste
- Definir a abordagem do teste de acordo com restrições do contexto
- Especificar tarefas e estimativas de prazos
- Algumas estratégias:
  - ◆ Analítica: baseada na análise de algum fator
  - ◆ Baseada em modelo: projetados com base em modelo de algum aspecto necessário do produto ( modelo de processo de negócio, de estado, de requisitos não funcionais )
  - ◆ Metódica: Conjunto pré-definido de testes, comparando as características de qualidade importantes

# Planejamento do teste

- Algumas estratégias:
- ◆ Compatível com processo: baseado em padrões definidos pela organização
  - ◆ Dirigida: orientado pelos stakeholders
  - ◆ Regressão: evitar regressão de recursos existentes
  - ◆ Reativo: é reativo ao componente ou sistema e aos eventos que ocorrem durante a execução

# Monitoramento e controle do teste

- Comparação contínua do progresso real com o plano de teste a partir de critérios de avaliação de saídas...ou seja, o **'done'**!
- Utilização de relatórios de progresso

# Análise do teste

- Base de teste é analisada de forma a analisar “o que testar” de acordo com critérios pré-estabelecidos
  - ◆ especificações de requisitos
  - ◆ documentos de arquitetura, fluxograma, casos de uso, etc...
  - ◆ código-fonte
- Avaliar os tipos de defeitos que podem ser encontrados
- Definir e priorizar condições de teste

# Modelagem do teste

- Responde a pergunta 'como testar'?
- As condições de teste são elaboradas em casos de teste de alto nível
- Priorização de casos de teste e conjuntos de casos de teste
- Verificar infraestrutura necessária e projetar ambiente de teste

# Implementação do teste

- Desenvolver e priorizar procedimentos de teste e possivelmente script automatizados
- Criar suítes de testes
- Organização lógica e eficiente da execução dos testes
- Preparar dados de teste



# Execução do teste

- Conjuntos de testes são executados conforme planejado, seja de forma manual ou automatizada
- Comparar resultados reais com resultados esperados
- Analisar anomalias para estabelecer prováveis causas
- Reportar e registrar essas anomalias
- Reteste

# Conclusão do teste

- Coletar dados das atividades de testes já concluídas de forma a revisar e consolidar a experiência
- Criar relatório de resumo de teste
- Finalizar e arquivar dados e registros dos testes
- Melhorar maturidade do processo de teste

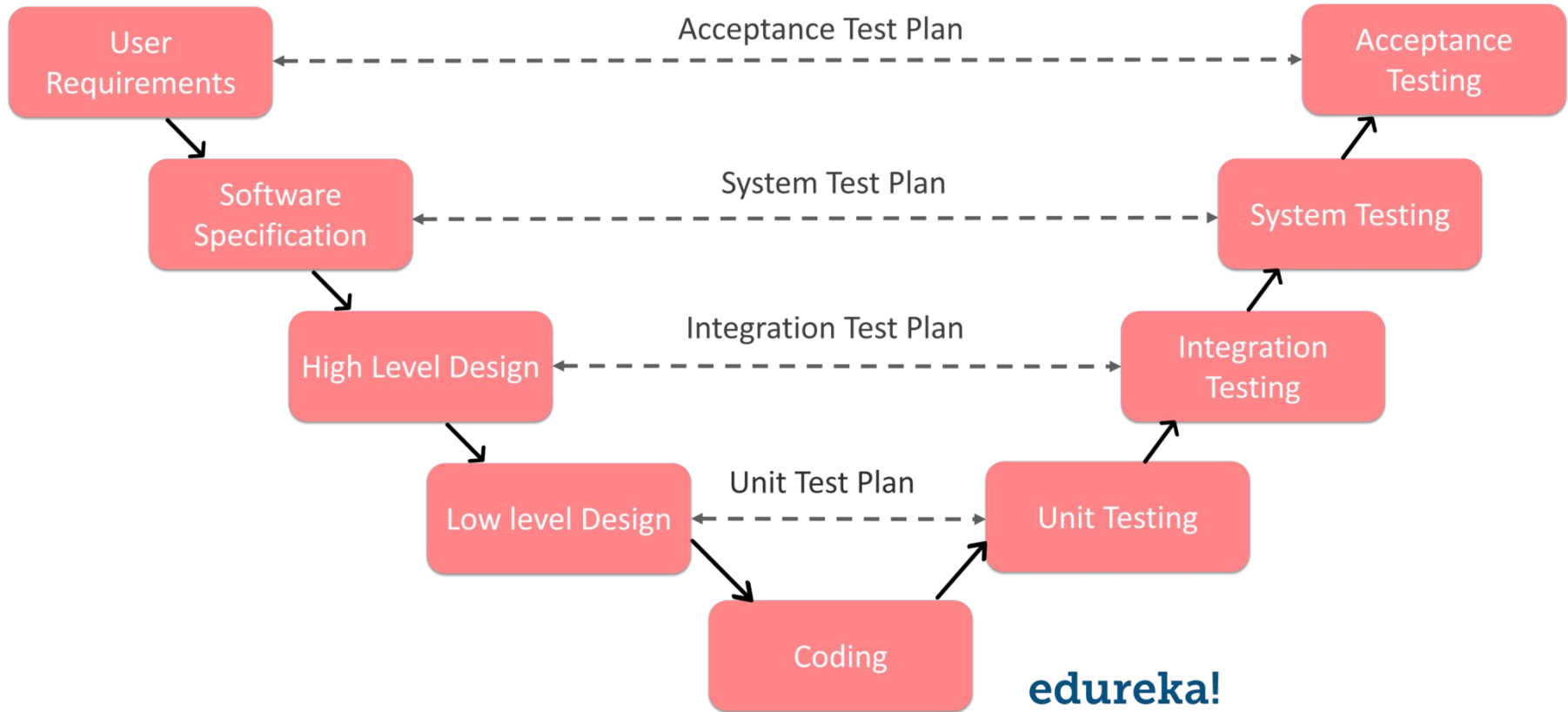
## Aula 3 . Etapa 3

# Níveis de teste

// Fundamentos de qualidade de software

# O que seriam esses níveis?

- São grupos de atividades de teste que são organizados e gerenciados juntos com relação ao nível de desenvolvimento
  - ◆ Teste de componentes
  - ◆ Teste de integração
  - ◆ Teste de sistema
  - ◆ Teste de aceite



# Testes de componente ou unidade

- Foco em testar componentes do código de forma independente
- Importante para:
  - ◆ Reduzir risco
  - ◆ Verificar requisitos funcionais e não-funcionais
  - ◆ Construir confiança do componente
  - ◆ Encontrar defeitos
  - ◆ Evitar que os defeitos sejam refletidos em níveis mais altos de teste

# Testes de integração

- Foco na integração entre componentes ou comunicação de sistemas
- Importante para:
  - ◆ Reduzir risco
  - ◆ Verificar interfaces
  - ◆ Encontrar defeitos nas partes envolvidas e sejam refletidos em níveis mais altos de teste

# Testes de sistema

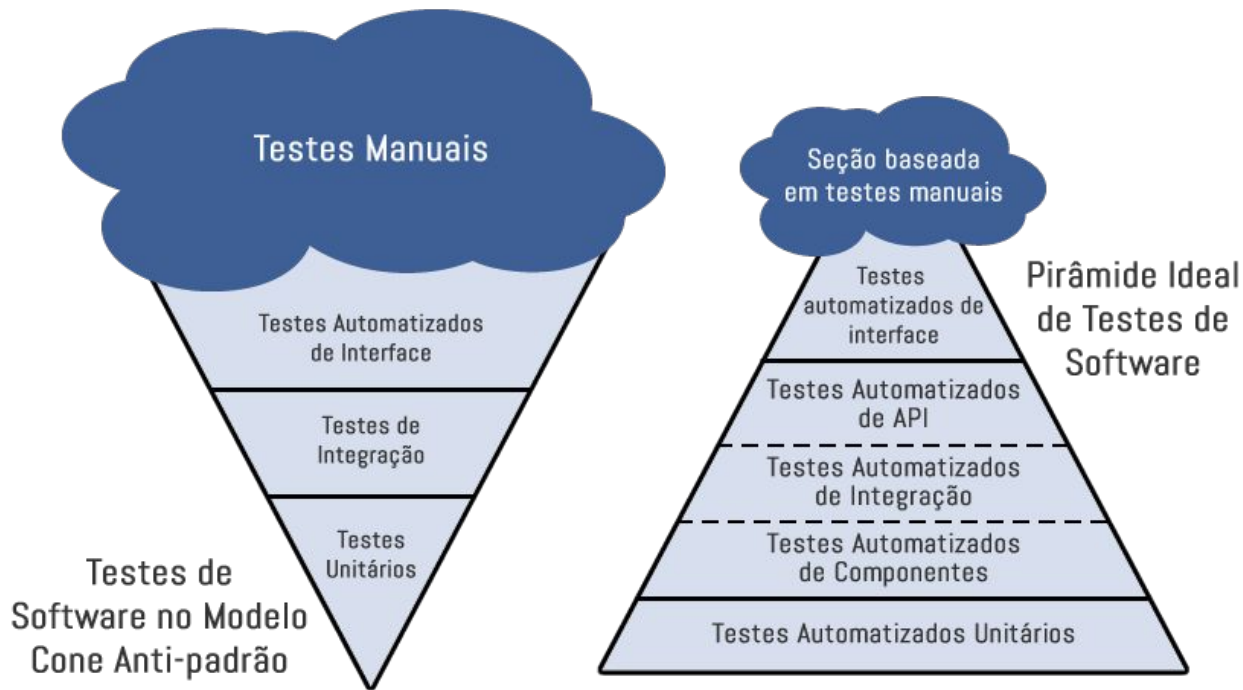
- Foco nos requisitos de ponta a ponta do sistema
- Importante para:
  - ◆ Reduzir risco
  - ◆ Validar sistema como um todo
  - ◆ Encontrar defeitos não vistos em níveis mais baixos
  - ◆ Evitar que defeitos se reflitam em produção após aceite do cliente



# Testes de aceite

- Foco nos requisitos de ponta a ponta do sistema do ponto de vista validação e conformidade com regras de negócio e necessidades do cliente
- Importante para:
  - ◆ Reduzir risco
  - ◆ Validar sistema como um todo
  - ◆ Encontrar defeitos não vistos em níveis mais baixos
  - ◆ Evitar que defeitos se reflitam em produção após aceite do cliente

# Pirâmide de testes



## Aula 3 . Etapa 4

# Tipos de teste

// Fundamentos de qualidade de software

# Tipos de teste e objetivos

- Grupo de atividades de teste destinado a verificar características específicas de um sistema, com base em objetivos específicos.
  - ◆ Avaliar características funcionais
  - ◆ Avaliar características não funcionais
  - ◆ Avaliar estrutura ou arquitetura de componente/sistema
  - ◆ Avaliar efeitos de alterações em outras partes do código

# Teste funcional

- Avaliação de funções que o sistema deve executar
- Desenvolvidos a partir de especificações de requisitos, histórias de usuário, casos de uso
- Os testes funcionais podem ser realizados em todos os níveis de teste
- Técnicas caixa-preta são bem úteis para avaliação de comportamentos funcionais do sistema

# Teste não funcional

- Avaliação de características não funcionais como usabilidade, eficiência de performance, segurança, etc...
- Também pode ser feito em todos os níveis de teste

# Teste caixa-branca

- Foco em testes com base na estrutura interna do sistema
  - ◆ Código-fonte
  - ◆ Arquitetura
  - ◆ Fluxo de dados
- Cobertura de código com testes de unidade ou integração

# Testes de mudanças

- Teste de confirmação: Verificação após defeito ser corrigido
- Teste de regressão: Verificação de efeitos colaterais nas alterações de um componente do sistema



## Aula 3 . Etapa 5

# Técnicas de teste

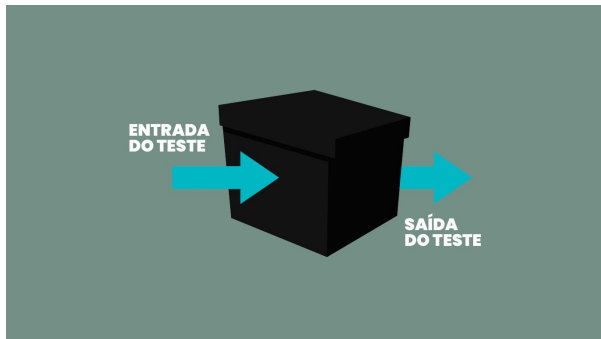
// Fundamentos de qualidade de software

# Objetivos das técnicas

- Auxílio na identificação das condições de teste, casos e seus dados
- Técnicas
  - ◆ Caixa-preta
  - ◆ Caixa-branca
  - ◆ Por experiência

# Técnicas de caixa-preta

- Fundamentadas em documentos de requisitos, casos de uso, histórias do usuário, etc...
- São aplicáveis para testes funcionais ou não-funcionais
- Foco nas entradas e saídas do teste, abstraindo a estrutura interna



1. **Particionamento de equivalência**
2. **Análise de valor limite**
3. **Tabela de decisão**
4. **Transição de estado**
5. **Caso de uso**

# Particionamento de equivalência

- Divide os dados em partições ou classes de equivalência que são processados da mesma forma, em formatos válidos e inválidos.

Exemplo: Um sistema de gestão e simulação de investimentos faz recomendações específicas dependendo da idade, tendo uma pontuação de risco de 0-100:

- até 18 anos : investimentos com risco 60-80
- 18 até 40 anos: investimentos com risco entre 40-60
- idade > 40 : investimentos com risco menor
- a idade máxima que o sistema faz previsões e simulações: 100 anos

Cenário para verificar simulação de investimentos

0 - 18

18 - 40

40 - 100

100 >

**Válidos**

**Inválidos**

# Análise de valor limite

- Estende o particionamento de equivalência quando a partição é ordenada e podemos analisar o valor mínimo e máximo

Cenário para verificar valor de frete

Exemplo: Um sistema de gerenciamento de envio de mercadorias possui as seguintes regras:

- O cliente não paga frete acima 100 reais
- Entre 50 e 100 reais, paga 20 reais
- Menor que 50 reais, o frete sobe para 35 reais
- Caso o valor total da compra chegue a R\$100.000 o cliente deve entrar em contato diretamente ou fazer uma nova compra.

0 - 50

50 - 100

100 - 100.000

0 <

100.000 >

**Válidos**

**Inválidos**

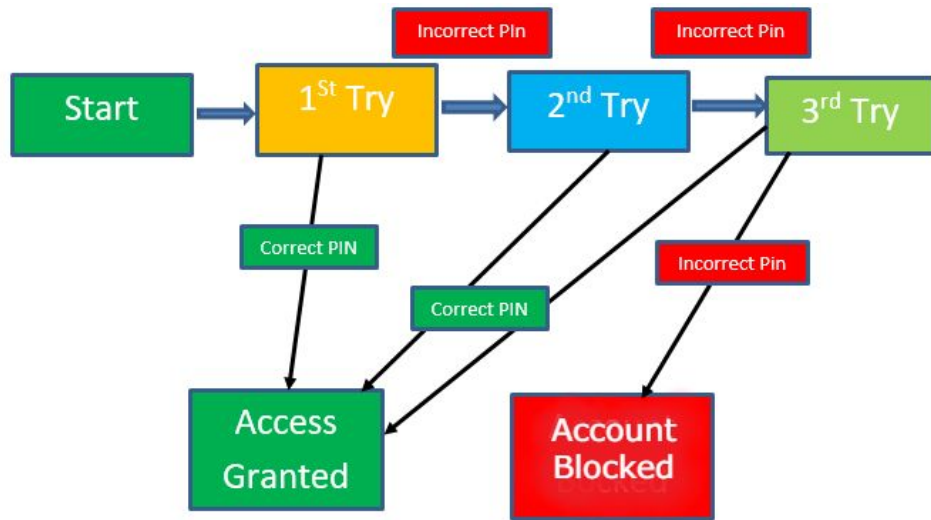
# Tabela de decisão

- Úteis para testar requisitos que especificam condições que combinações com diferentes resultados

Variáveis	1	2	3	4
Cartão válido?	Não	Sim	Sim	Sim
Senha Válida?	X	Não	Sim	Sim
Valor solicitado é <= Saldo	X	x	Não	Sim
Saída Esperada	Cartão Inválido	Senha inválida	Saldo Insuficiente	Saque efetuado com sucesso

# Transição de estado

- Situações em que o sistema reage diferente a um evento dependendo das condições atuais ou de um histórico, que pode ser resumido como estados



Fonte: Guru99(2022)

# Transição de estado

→ É gerada uma tabela de transição que vai direcionar os casos de teste

	Correct PIN	Incorrect PIN
S1) Start	S5	S2
S2) 1 <sup>st</sup> attempt	S5	S3
S3) 2 <sup>nd</sup> attempt	S5	S4
S4) 3 <sup>rd</sup> attempt	S5	S6
S5) Access Granted	–	–
S6) Account blocked	–	–

Fonte: Guru99(2022)



# Teste de caso de uso

- Derivados naturalmente dos casos de uso
- Associa-se ações com os atores do caso
- Projeta-se testes para casos básicos, alternativos e de erros

# Técnicas de caixa-branca

- Baseadas na estrutura interna do objeto de teste
- Podem ser usadas em todos os níveis de teste
- Normalmente usada para testes a nível de componente no código-fonte

1. Cobertura de instruções
2. Cobertura de decisões



# Teste de cobertura de instruções

- Testa instruções executáveis do código
- Cobertura medida como (número de instruções executadas)/ (total de instruções)



[Cobertura de código explicada. Relatórios e métricas com Istanbul e o... | by Eduardo Rabelo | Medium](#)

# Teste de cobertura de decisões

- Testa as condicionais existentes no código e o que é executada em cada decisão
- Cobertura = número de resultados de decisão executados/ total de resultados de decisão no objeto
- 100% de cobertura de decisão → 100% de cobertura de instrução

# Teste de cobertura de decisões

- Testa as condicionais existentes no código e o que é executada em cada decisão
- Cobertura = número de resultados de decisão executados/ total de resultados de decisão no objeto
- 100% de cobertura de decisão → 100% de cobertura de instrução

# Técnicas baseadas na experiência

- Baseada em experiência e intuição de quem testa
  - Pode-se identificar situações não encontradas nos métodos mais sistemáticos
  - Cobertura de difícil avaliação e medição
- 
1. **Suposição de erro**
  2. **Teste exploratório**
  3. **Baseado em checklist**

# Fundamentos de Qualidade de Software

Carolina Santana Louzada

Engenheira de Qualidade de Software - UOLEdtech

# Para saber mais

- ★ [CTFL \(bstqb.org.br\)](http://bstqb.org.br)
- ★ Família SQuaRE: ISO/IEC 25000-25099
- ★ [martinfowler.com](http://martinfowler.com)
- ★ [Cobertura de código explicada. Relatórios e métricas com Istanbul e o... | by Eduardo Rabelo | Medium](#)



# Percurso

## Aula 1

O que é qualidade de software?

## Aula 2

Gerenciamento de defeitos

## Aula 3

Introdução aos testes de software

# Dúvidas durante o curso?

- > Fórum do curso
- > Comunidade online (Discord)



SCAN ME