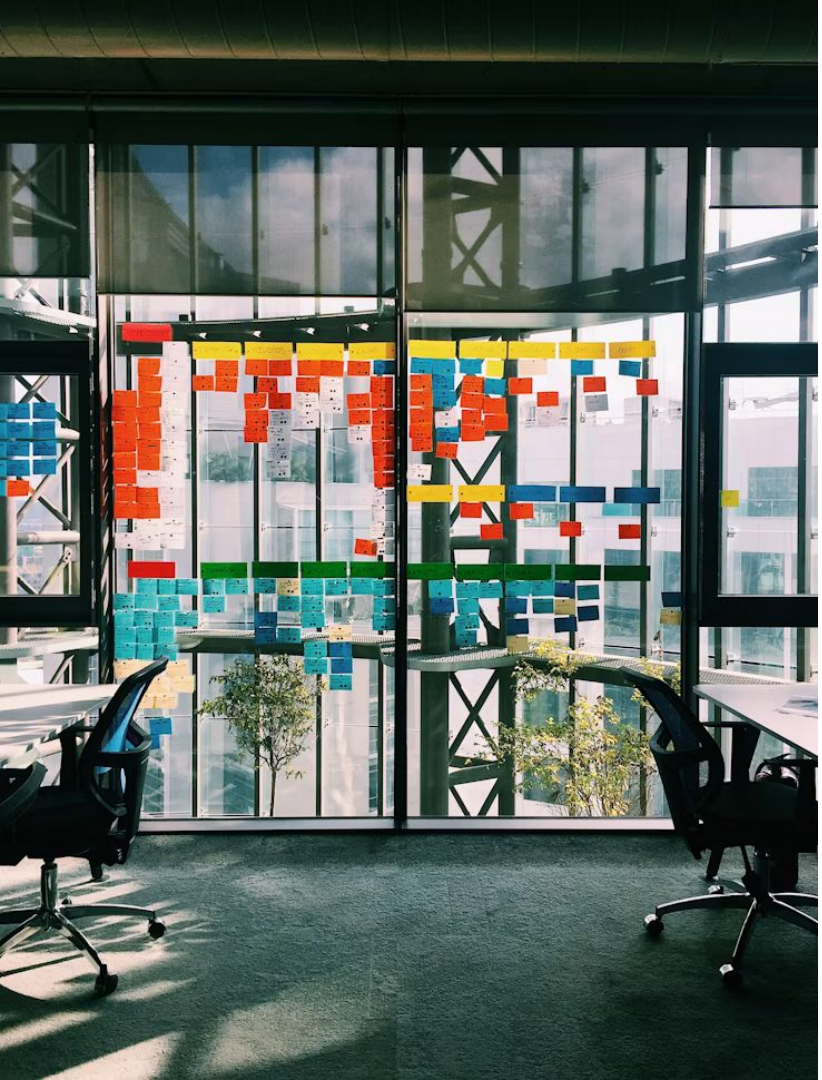


5954014 - Engenharia de Software

Aula 14 - Teste de Software

Prof. Dr. Denis Martins

DCM | FFCLRP | USP

Duas reservas, uma sala: o que faltou testar?

A mesma sala foi confirmada para dois usuários em horários sobrepostos.



Como identificar e tratar problemas como este antes de enviar o código para produção (deploy)?

Objetivos de Aprendizagem

01	Explicar a contribuição dos testes para verificação e validação.
02	Diferenciar níveis de teste, técnicas de seleção e objetivos dos testes.
03	Identificar tipos de teste.
04	Derivar casos de teste a partir de requisitos.
05	Interpretar cobertura de instruções e de decisões.



Testar produz evidências

Teste de software é a execução planejada do software, com condições e expectativas definidas, para revelar falhas e avaliar seu comportamento.

01 1. Preparar

Definir previamente as pré-condições, dados de entrada e o estado inicial do sistema.

02 2. Executar

Rodar o software sob as condições planejadas e coletar as respostas reais do sistema.

03 3. Comparar

Confrontar o resultado obtido com o resultado esperado para identificar inconsistências.

Exemplo Prático: Tentativa de agendamento de **29 minutos** (abaixo do limite mínimo) → **Ação:** Solicitar reserva → **Resultado esperado:** A solicitação deve ser rejeitada pelo sistema.

Limitação do Teste: Testes aprovados **aumentam a confiança** nos comportamentos exercitados; contudo, eles **não demonstram a ausência de defeitos** no software como um todo.

Verificação e Validação

Verificação

Foco na Especificação

O produto atende às especificações?

Exemplo Prático

A reserva respeita o limite de 120 minutos?

Validação

Foco na Necessidade

O produto atende às necessidades de uso?

Exemplo Prático

O limite atende a uma atividade acadêmica de três horas?

Síntese: Cumprir a especificação não garante que ela represente a necessidade correta.

Níveis de Teste: Qual é o Alcance?

Nível	Foco	Exemplo Prático
Unidade	Unidade de comportamento	Validar duração.
Integração	Interação entre partes	Armazenar e recuperar com banco de testes.
Sistema	Sistema completo	Solicitar pelo fluxo completo.
Aceitação	Adequação ao uso	Usuários avaliam critérios acordados.

Importante: Aceitação destaca a decisão dos interessados. Não é sinônimo de teste de release.

O que queremos avaliar?

Funcional

Rejeitar participantes acima da capacidade.

Desempenho

Medir resposta sob carga definida.

Estresse

Avaliar além da carga prevista.

Segurança

Impedir acesso indevido às reservas de outro usuário.

Exemplo didático: “**95%** das consultas em até **2 s**, com **100** usuários simultâneos”.

De onde vêm os casos de teste?

Caixa-preta

Selecionar casos a partir da **especificação**.

Exemplo de Comparação
“Aceitar 120 minutos”

Caixa-branca

Selecionar casos a partir da **estrutura interna**.

Exemplo de Comparação
“Exercitar resultados de uma decisão”

Caixa-cinza
Usar conhecimento parcial da implementação.

Destaque: Um teste de unidade pode usar técnica de caixa-preta.

Das regras aos resultados esperados

ID	Regra resumida (Oráculo / Regras didáticas do caso)
R1	Autenticação obrigatória.
R2	Duração inteira: 30–120 minutos, **inclusive**.
R3	Participantes: inteiro positivo até a capacidade.
R4	Sem sobreposição; horários **adjacentes** permitidos.
R5	Armazenar e apresentar na consulta.

Notas de Aplicação / Exercícios:

- Início é **incluído** e término é **excluído** no intervalo de tempo.
- As regras acima servem como o **oráculo** oficial do exercício.
- Para testar uma regra específica, mantenha todas as demais condições válidas.

Rastreabilidade

Relacionar cada **caso de teste** ao seu respectivo **requisito** correspondente.

Oráculo

Mecanismo ou referência utilizada para determinar o **resultado esperado** do teste.

Especificar um caso de teste

Campo	CT-01
Requisito e objetivo	**R2:** aceitar a duração mínima.
Pré-condições	Usuário autenticado; sala disponível; capacidade respeitada.
Dados	Início 10h; término 10h30.
Ação	Solicitar a reserva.
Resultado esperado	**Confirmar, armazenar e mostrar na consulta.**
Observado / situação	*Preencher após executar.*

Nota de Aplicação:

- "Armazenar" e "consultar" também se vinculam a **R5**.
- Um único caso de teste pode se vincular a múltiplos requisitos, desde que as verificações fiquem claramente identificáveis.

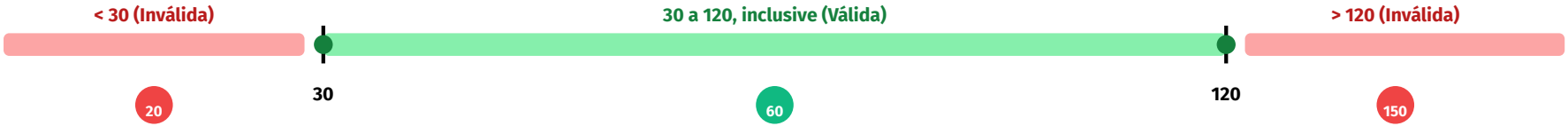
Pergunta para Reflexão

Por que "funcionou corretamente" é insuficiente como resultado esperado?

- **Ambiguidade:** Não define o critério objetivo de aprovação/falha.
- **Subjetividade:** Testadores diferentes podem ter interpretações distintas.
- **Automação:** Scripts e oráculos de teste exigem validações precisas e mensuráveis.
- **Oráculo:** As expectativas devem sempre anteceder a execução.

Particionamento e Equivalência

Objetivo: Dividir o domínio de entrada em classes para as quais se espera comportamento semelhante, selecionando representantes para o teste.



Marcadores indicam os valores representantes testados em cada classe

Partição de duração inteira	Representante	Resultado Esperado
Menor que 30	20	Rejeitar
Entre 30 e 120, inclusive	60	Aceitar
Maior que 120	150	Rejeitar

Valores-limite: testar nas fronteiras

Objetivo: Selecionar valores na fronteira e imediatamente ao redor dela, sensíveis a defeitos de comparação.

Fronteira Inferior (Limite: 30)



Exemplo de Defeito:
Trocar **>** por **≥** ou **<** por **≤** pode alterar incorretamente a aceitação do limite.

Fronteira Superior (Limite: 120)



Atenção ao Resultado Esperado:
O resultado esperado vem sempre da regra de negócio. Justifique rigorosamente a inclusão de 120.

Atividade: projete oito casos

> **Objetivo:** aplicar as regras na especificação de testes.

Para cada caso: requisito, pré-condições, dados, ação e resultado esperado.

Quantidade	Situações
4	Duração de 29, 30, 120 e 121 minutos.
2	Sala de capacidade 20: solicitar para 20 e 21 participantes.
2	Reserva existente 10h–11h: solicitar 10h30–11h30 e 11h–12h.

Automatizar: preparar, executar e verificar

1. Preparar

Definir estado e dados de entrada necessários para o teste.

2. Executar

Chamar a operação ou ação que está sendo testada.

3. Verificar

Comparar o resultado observado com o resultado esperado.

Asserção
Verificação executável de uma expectativa.

Suíte
Conjunto de testes organizados.

Importante: Automação repete verificações. Selecionar casos exige análise.

Do caso escrito à asserção

Mapear as etapas do caso de teste para uma verificação executável.

1. Preparar
2. Executar
3. Verificar

```python
def validar_duracao(duracao: int) -> bool:
    return 30 <= duracao <= 120


def test_aceita_duracao_maxima():
    # 1. Preparar
    duracao = 120

    # 2. Executar
    resultado = validar_duracao(duracao)

    # 3. Verificar
    assert resultado is True
```

Teste de Unidade vs. Teste de Sistema

Este teste de unidade verifica **somente a duração**. Diferentemente do caso de sistema, ele não realiza verificações de armazenamento no banco ou consulta de dados.

Onde exatamente está localizada a expectativa executável no código acima?

Do caso escrito à asserção: Outro Exemplo

1. Preparar

2. Executar

3. Verificar

```
import unittest

def validar_duracao(duracao: int) -> bool:
    return 30 <= duracao <= 120

class TestValidacaoDuracao(unittest.TestCase):

    def test_aceita_duracao_maxima(self):
        # 1. Preparar
        duracao = 120

        # 2. Executar
        resultado = validar_duracao(duracao)

        # 3. Verificar
        self.assertTrue(resultado)

if __name__ == "__main__":
    unittest.main()
```

Exercício: onde está a divergência?

Implementação Recebida

```
# Função sob teste
função validarDuracao(d):
  se d < 30:
    retornar falso
  se d ⩾ 120:
    retornar falso
  retornar verdadeiro
```

Tabela de Execução

Duração	Esperado	Observado
30	verdadeiro	____
60	verdadeiro	____
120	verdadeiro	____

Análise Crítica & Discussão:

“ual condição da função acima está em desacordo com a Regra R2

O que torna um teste confiável?

Propriedades do acrônimo FIRST para testes de unidade eficazes

F: Fast
Rápido na execução.

I: Independent
Independente de outros testes.

R: Repeatable
Repetível em qualquer ambiente.

S: Self-checking
Autoverificável (passa ou falha).

Sinais de Alerta e Problemas Comuns

Dependência de Estado

“Só passa depois de outro teste”

Indica violação do princípio de independência (I). Testes devem ser isolados e não compartilhar estado executável.

Instabilidade Temporal

“Muda de resultado conforme o horário”

Indica violação do princípio de repetibilidade (R). Causa testes intermitentes devido a dependências temporais não controladas.

Testabilidade e controle de dependências

Testabilidade: facilidade de preparar condições, executar e observar resultados.

Serviço de Reserva

Dominio ---- **Interface**

Isolamento do domínio via interfaces permite alternar simuladores e infraestrutura real.

Opção A: Ambientes Simulados

Stub / Mock

Respostas pré-definidas ou verificação de chamadas diretas sem custo de E/S.

Opção B: Integração Real

Banco de Testes

Exercita persistência, gravação e leitura reais para validar o meio de integração.

Recurso	Uso no Teste
Stub	Resposta preparada: sala disponível ou ocupada.
Mock	Verificar uma interação esperada, como notificar uma confirmação.
Dependência real	Exercitar gravação e leitura no banco de testes.

Simular uma dependência não comprova que a integração real funciona.

Cobertura: Critérios estruturais e sua contagem para avaliação de testes

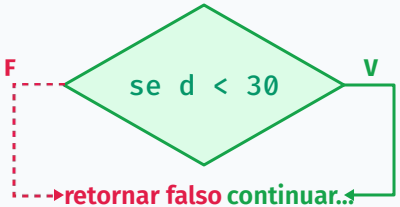
Cobertura de Instruções

$$\text{Cobertura} = \frac{\text{instruções executadas}}{\text{instruções consideradas}} \times 100\%$$

Cobertura de Decisões

$$\text{Cobertura} = \frac{\text{resultados exercitados}}{\text{resultados possíveis}} \times 100\%$$

Exemplo Prático: Múltiplos Resultados



Cada condição de decisão simples possui exatamente **dois resultados possíveis** (Verdadeiro ou Falso).

- Uma decisão binária simples gera **2 resultados possíveis**.
- Duas decisões binárias simples acumulam **4 resultados possíveis** no total.

Importante: Cobertura mede o que foi executado. Ela não demonstra a correção do código.

100% de cobertura, um defeito presente

Código Submetido a Teste

```
// Função de validação
if (d < 30) {
    return falso;
}
if (d >= 120) { // defeito!
    return falso;
}
return verdadeiro;
```

Suíte de Testes Executada

Entrada	d < 30	d >= 120	Retorno
20	V	Não executada	falso
60	F	F	verdadeiro
150	F	V	falso

✓ Todas as instruções exercitadas.
✓ Decisões: 4/4 resultados = 100% de cobertura.

Pergunta: Qual valor permitido ainda é rejeitado pelo código?

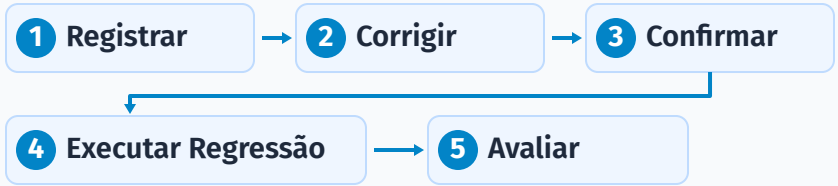
Após Mudar: Confirmar e Procurar Regressões

Ações de Reexecução

Reexecutar 120 minutos
Confirmar a correção específica.

Reexecutar outros casos
Procurar regressões.

Fluxo do Processo



Campos Mínimos para Registro de Falhas

Contexto / Versão
Ambiente e build

Pré-condições
Estado inicial

Passos e Dados
Ações executadas

Resultado Esperado
Comportamento correto

Resultado Observado
Falha ocorrida

Evidência
Logs e capturas

Teste de Software — Resumo e Conclusão | Da especificação à confiança

1 Definir o esperado
Derivar os testes de requisitos claros e verificáveis.

2 Selecionar bons casos
Combinar partições, valores-limite e critérios estruturais.

3 Testar em diferentes níveis
Unidade, integração, sistema e aceitação.

4 Automatizar e repetir
Usar testes confiáveis para obter feedback e detectar regressões.

5 Interpretar as evidências
Considerar resultados, limitações e riscos antes da liberação.

Testes aprovados e alta cobertura aumentam a confiança, mas não demonstram ausência de defeitos.

“O que nossos testes verificaram — e quais riscos ainda permanecem?”

Resumo e Conclusão

1 Definir o esperado
Derivar os testes de requisitos claros e verificáveis.

2 Selecionar bons casos
Combinar partições, valores-limite e critérios estruturais.

3 Testar em diferentes níveis
Unidade, integração, sistema e aceitação.

4 Automatizar e repetir
Usar testes confiáveis para obter feedback e detectar regressões.

5 Interpretar as evidências
Considerar resultados, limitações e riscos antes da liberação.

Testes aprovados e alta cobertura aumentam a confiança, mas não demonstram ausência de defeitos.

Referências

Bibliografia Básica da Disciplina

Roger S. Pressman & Bruce R. Maxim

Engenharia de Software: Uma Abordagem Profissional

Capítulos 19 e 20 (completos)

Ian Sommerville

Engenharia de Software

Capítulo 8 (completo)

Referências Complementares de Apoio

Marco Tulio Valente
Engenharia de Software Moderna
Capítulo 8 (completo)
https://engsoftmoderna.info/cap8.html

Can VIBE Drive Development? Threats and Opportunities
https://cacm.acm.org/opinion/can-vibe-drive-development-threats-and-opportunities/

Dúvidas e Discussão