Engenharia de Software

Qualidade de Software

Profª Raquel Mini

raquelmini@ufmg.br

DEE / UFMG

Unidade V

- Qualidade de Software
  - Gerenciamento de qualidade
  - Verificação e validação
    - Técnicas de revisão
    - Testes de software

O que é qualidade?

- Crosby fornece uma resposta singular para essa questão:
  - "O problema de gestão da qualidade não é o que as pessoas não sabem a respeito dela. O problema é o que elas pensam que sabem...”
    - Todo mundo é a favor
      - Sob certas circunstâncias, certamente
    - Todo mundo se considera um entendido do assunto
      - Mesmo que não queiram explicá-lo
    - Todo mundo pensa que a execução é apenas uma questão de seguir as inclinações naturais
      - Apesar de tudo conseguimos de alguma forma
    - E, certamente, a maioria das pessoas acha que problemas nessas áreas são causados pelos outros...

O que é qualidade de software?

- Conjunto de características que devem ser alcançadas em um determinado grau para que o produto atenda às necessidades de seus usuários
- Totalidade de características de uma entidade que lhe confere a capacidade de satisfazer a necessidades explícitas e implícitas
- Conformidade a:
  - Requisitos funcionais e de desempenho
  - Padrões e convenções de desenvolvimento pré-estabelecidos
  - Atributos implícitos que todo software desenvolvido profissionalmente deve possuir

O que é qualidade de software?

- O grau no qual um sistema, componente ou processo satisfaz os requisitos especificados e as necessidades e expectativas do cliente ou usuário (IEEE)
- Atingir excelentes níveis de adequação ao uso (Watts Humphrey)
- Qualidade dirigida por mercado – atingir satisfação total do cliente (IBM)
- Aderência ao uso (Joseph Juran)

Realidade do mercado atual de software

- Software tornou-se parte da vida quotidiana
  - Considerada por alguns autores a “tecnologia individual mais importante no palco mundial”
  - Entrega o produto mais importante na nossa época: informação
  - Alterou (e continua alterando) os hábitos de cada vez mais pessoas da sociedade

Ariane 5 (1996)

- Código importado da Ariane 4

- *Overflow* na conversão de ponto flutuante
- Prejuízo de USD 500 milhões

Iphone 4 (2010)

- Dificuldades de ligar ou receber ligações devido à falha de isolamento da antena
- Milhões de clientes afetados
- Solução: capa para o aparelho
- Impactos significativos
  - Imagem
  - Risco de recall

Caos aéreo GOL (2010)

- Atualização do software para planejamento e escala da tripulação
- Geração de dados incorretos que resultaram no "planejamento inadequado da malha aérea e da jornada de trabalho dos tripulantes"
- Consequências
  - Imagem
  - Custos
  - Pedido de desculpas público do presidente da companhia

Qualidade de software

- A qualidade de software tem se aprimorado nos últimos 15 anos
  - Empresas têm adotado novas técnicas
  - Orientação a objetos tem se difundido
  - Ferramentas CASE têm sido usadas
- Na manufatura, qualidade significa atender às especificações
  - Em software, a definição não é tão simples

Qualidade de software

- Não é fácil definir qualidade de software como adequado à especificação
  - É difícil escrever especificações de software completas e precisas
  - A especificação pode estar ambígua, incompleta ou inconsistente
  - Alguns requisitos podem não aparecer na especificação
  - É difícil especificar todos os requisitos

Qualidade de software

- Alguns atributos são difíceis de serem especificados
  - Mas tem grande efeito na qualidade do sistema

- Exemplos
  - Como garantir segurança dos dados?
  - Como documentar sobre eficiência?
  - Como especificar a facilidade de manutenção?

Qualidade de software

- Pode ser impossível concluir objetivamente se um sistema de software cumpre ou não suas especificações
  - Exemplo: é difícil avaliar os atributos de qualidade de software, como manutenibilidade, sem usar o software por um longo período

Alguns atributos de qualidade

Segurança	Compreensibilidade	Portabilidade
Proteção	Testabilidade	Usabilidade
Confiabilidade	Adaptabilidade	Reusabilidade
Resiliência	Modularidade	Eficiência
Robustez	Complexidade	Capacidade de aprendizado

Qualidade do processo e do produto

- Manufatura
  - Existe uma clara ligação entre a qualidade do processo e do produto uma vez que o processo é relativamente fácil de ser padronizado e monitorado
- Software (é criado e não manufaturado)
  - O desenvolvimento de software é um processo criativo, em vez de mecânico, portanto, a influência de competências e experiências individuais é significativa
  - Fatores externos (novidade de uma aplicação ou pressão comercial) também afetam a qualidade do produto, independentemente do processo usado

Qualidade do processo e do produto

- Não há dúvida de que o processo de desenvolvimento usado tenha uma influência significativa sobre a qualidade de software
  - Bons processos são mais suscetíveis de produzir o software de boa qualidade

Equipe de qualidade

- Idealmente, a equipe de garantia da qualidade deve ser diferente da equipe de desenvolvimento
- O processo de qualidade envolve
  - Definir padrões de processo
  - Monitorar o processo para verificar o adequado uso dos padrões
  - Emitir relatórios para a gerência de projeto e da organização

O tamanho do projeto

- Mesmo em projetos pequenos, o gerenciamento da qualidade é importante
  - Entretanto, ele pode seguir uma abordagem mais informal
- Em sistemas grandes, o gerenciamento de qualidade requer três atividades
  - Garantia de Qualidade
  - Planejamento de Qualidade
  - Controle de Qualidade

Atividades do gerenciamento da qualidade

- Garantia de qualidade
  - Definição de processos e padrões que devem conduzir a produtos de alta qualidade
- Planejamento de qualidade
  - Seleção dos procedimentos e padrões apropriados ao projeto
- Controle de qualidade
  - Aplicação dos processos de qualidade visando eliminar os produtos que não atingiram o nível de qualidade exigido
  - Realização de verificação e validação (V&V)

Verificação e Validação (V&V)

Verificação e validação

- Objetivos da verificação e validação
  - Mostrar que o software atende à sua especificação
  - Mostrar que o software atende às necessidades do cliente

Verificação

- O objetivo é verificar se o software atende aos requisitos funcionais e não funcionais especificados
- Verificação inclui a realização de testes para encontrar erros
- Pergunta principal
  - *Estamos construindo o produto corretamente?*

Validação

- A inexistência de erros não mostra a adequação operacional do sistema
  - Deve ser feita a **validação** com o cliente
- A validação procura assegurar que o sistema atenda as expectativas e necessidades do cliente
- Pergunta principal
  - *Estamos construindo o produto correto?*

Verificação e validação (V&V)

- Técnicas de V&V
  - Revisões e inspeções
  - Testes de software (principal técnica de V&V)

Técnicas de Revisão

Revisões de software

- São como um “filtro” para a gestão de qualidade
- São aplicadas em várias etapas durante o processo de engenharia de software
- Servem para revelar erros e defeitos que podem ser eliminados
- Elas purificam o resultado

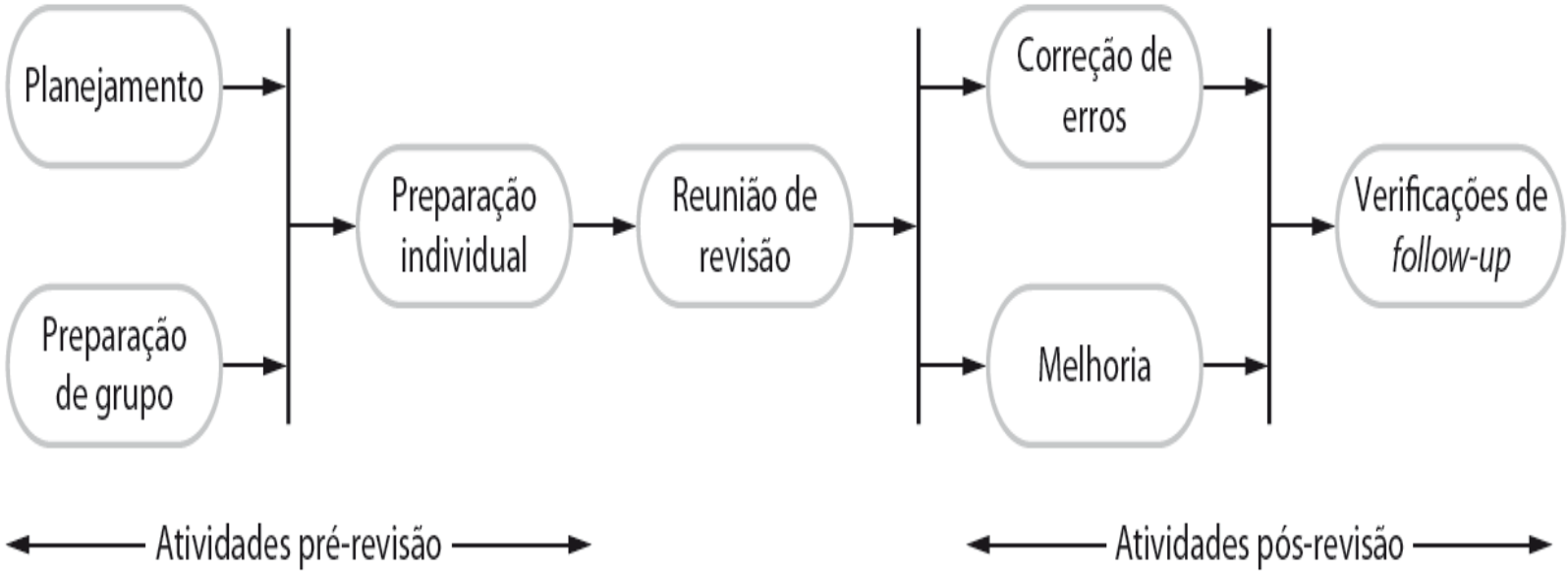
Revisões e inspeções

- São atividades de controle de qualidade que verificam a qualidade dos entregáveis de projeto
  - Examinam o software, sua documentação e os registros do processo para descobrir erros e omissões e verificar se os padrões de qualidade foram seguidos
- O objetivo é melhorar a qualidade de software e não avaliar o desempenho das pessoas na equipe de desenvolvimento

Revisões

- Um grupo de pessoas examina o software e a documentação associada à procura de possíveis problemas e não conformidade com padrões
- Deve verificar a consistência e a completude dos documentos ou código em revisão e certificar-se de que os padrões de qualidade foram seguidos
- Ajudam a descobrir problemas e omissões no software ou na documentação de projeto
- As conclusões das revisões devem ser formalmente registradas como parte do processo de gerenciamento de qualidade

Processo de revisão de software



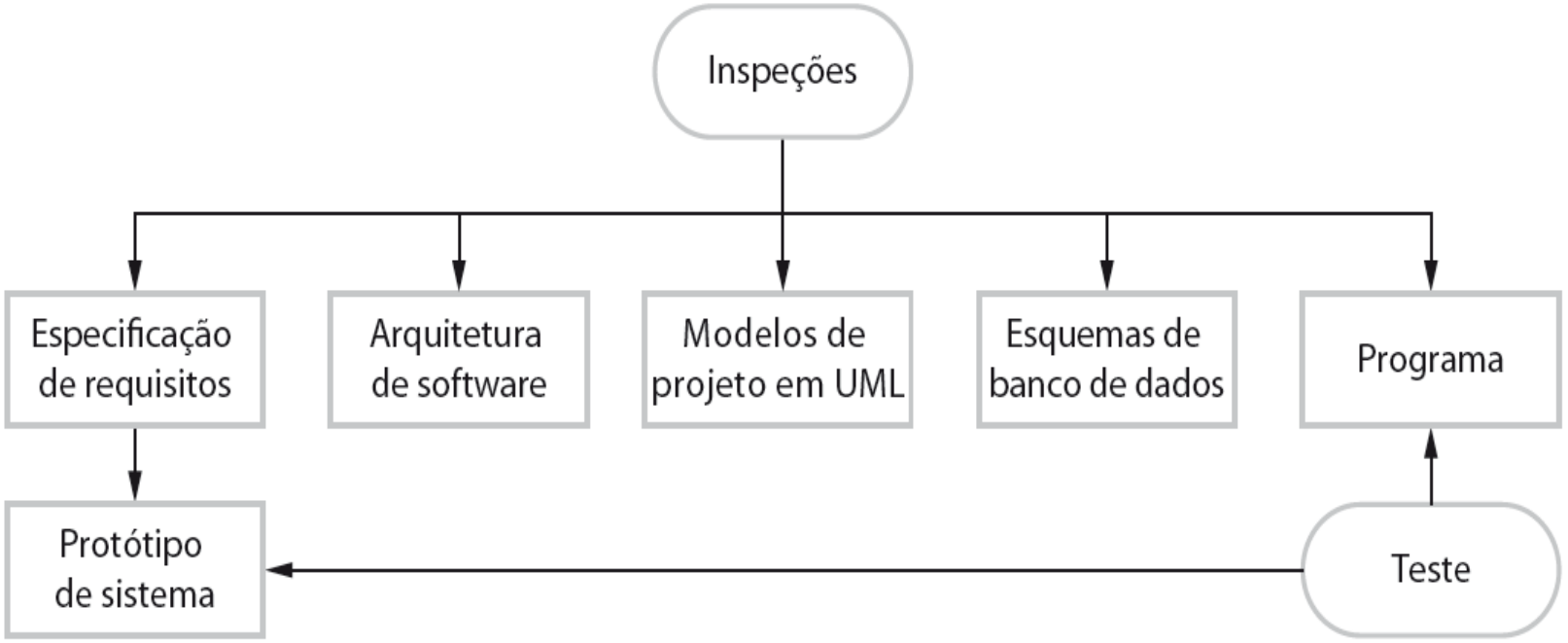
Inspeções

- São revisões em pares em que os membros da equipe colaboram para encontrar *bugs* no programa que está sendo desenvolvido
- Técnica estática de V&V
  - Não precisa executar o programa
- Pode ser usada em qualquer atividade de desenvolvimento
  - Requisitos, projeto, código fonte, etc.
- Pode ser semi-automatizada por análise estática
  - Análise não automatizada é cara

Testes x inspeção de software

- Teste é uma técnica dinâmica de V&V
  - Consiste em executar uma implementação com dados de teste
- Pode ser usada para avaliar os requisitos não funcionais
  - Desempenho, confiabilidade, segurança, etc.

Verificação estática e dinâmica



Vantagens da inspeção

- Muitos defeitos diferentes podem ser descobertos em uma única inspeção
  - Um teste revela um defeito e oculta vários
- Versões incompletas do sistema podem ser inspecionadas
- Permite encontrar problemas em outros atributos de qualidade do software
  - Uso de um algoritmos mais eficiente, padrão de projeto, etc.

Limitações da inspeção

- Inspeção não é adequada
  - Para descobrir defeitos nas interações
  - Para demonstrar se o software é útil
  - Para verificar requisitos não funcionais, como desempenho, segurança, etc.
- Inspeção é uma técnica cara
- Ela não permite validar com o cliente
  - Somente verifica a correspondência entre a especificação e o software

Exercícios

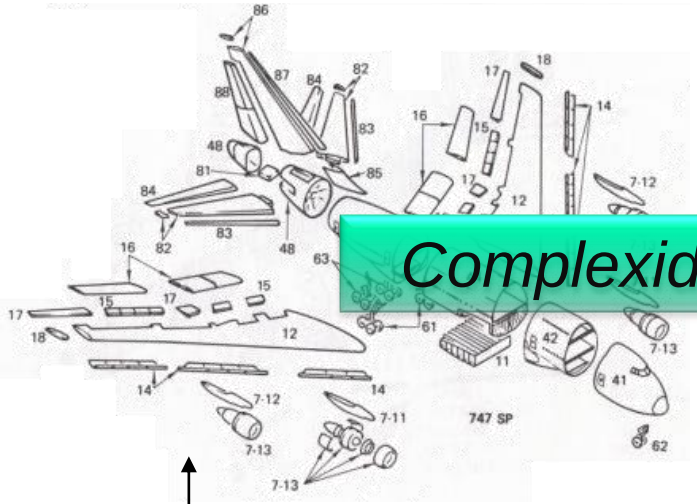
1. Explique por que inspeções de programa são uma técnica eficaz para descobrir erros em um programa. Que tipos de erros são improváveis de serem descobertos por meio de inspeções?

Testes de Software

O que são testes de software?

- Executar um programa ou parte de um programa com a intenção de encontrar erros (Myers)
- Executar um sistema ou componente sob condições controladas para detectar diferenças entre os resultados obtidos e os esperados (SWEBOK)
- Medir a qualidade de um sistema
- Demonstrar a presença de defeitos, e nunca sua ausência (Dijkstra)

Por que testar?



Complexidade

Prazo

Custos

Qualidade

38

O contra-exemplo

- “Acredito que o último bug do TeX foi encontrado e removido em 27 de Novembro de 1985. Entretanto, de toda forma, se houver ainda algum erro furtivo no código, eu alegremente pagarei o valor de US$20,40 à primeira pessoa que o descobrir (isso é duas vezes o valor anterior e eu planejo dobrá-lo em um ano; perceba, eu realmente estou seguro)”
  - (Donald E. Knuth) – Prefácio de Tex: The Program
  - Knuth definiu que o valor poderia continuar aumentando (dobrando) até o valor máximo de US$327.68
  - Em uma aula em 2002, Knuth disse “Não existe erro reportado para o TeX desde 1994 ou 1995”

Desafios

- Software cada vez mais utilizado pelo público geral
  - Celulares, PADs, veículos, dispositivos “inteligentes”
- Testes em ambientes mais críticos
  - Hospitalar, aeronáutica, instituições bancárias
- Aspectos não-funcionais
  - segurança, desempenho, latência
- Necessidade de conhecimento
  - Negócio
  - Tecnologia
  - Técnicas

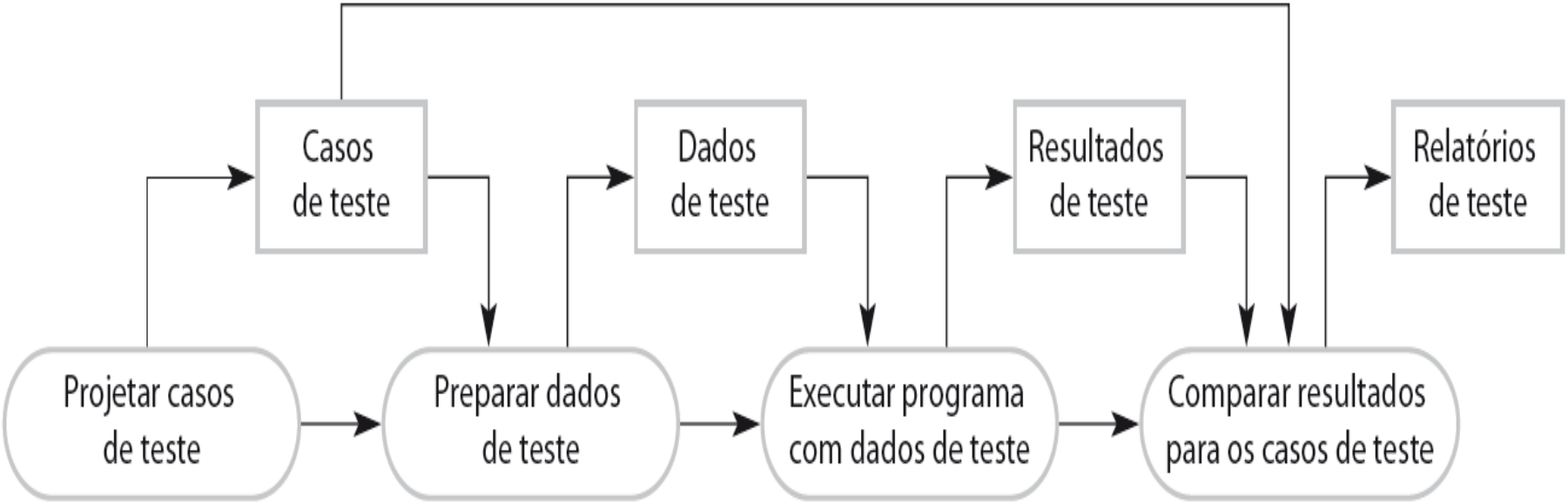
Testes de Software

- Testes chegam a ocupar 50% do tempo de projeto

- Testes fazem parte do processo de verificação e validação (V&V)
  - Devem ser usados em conjunto com a verificação estática (inspeção)

Processo de teste

- Exemplo de processo de teste baseado em planos
  - As atividades são planejadas antes de serem executadas



Casos e dados de teste

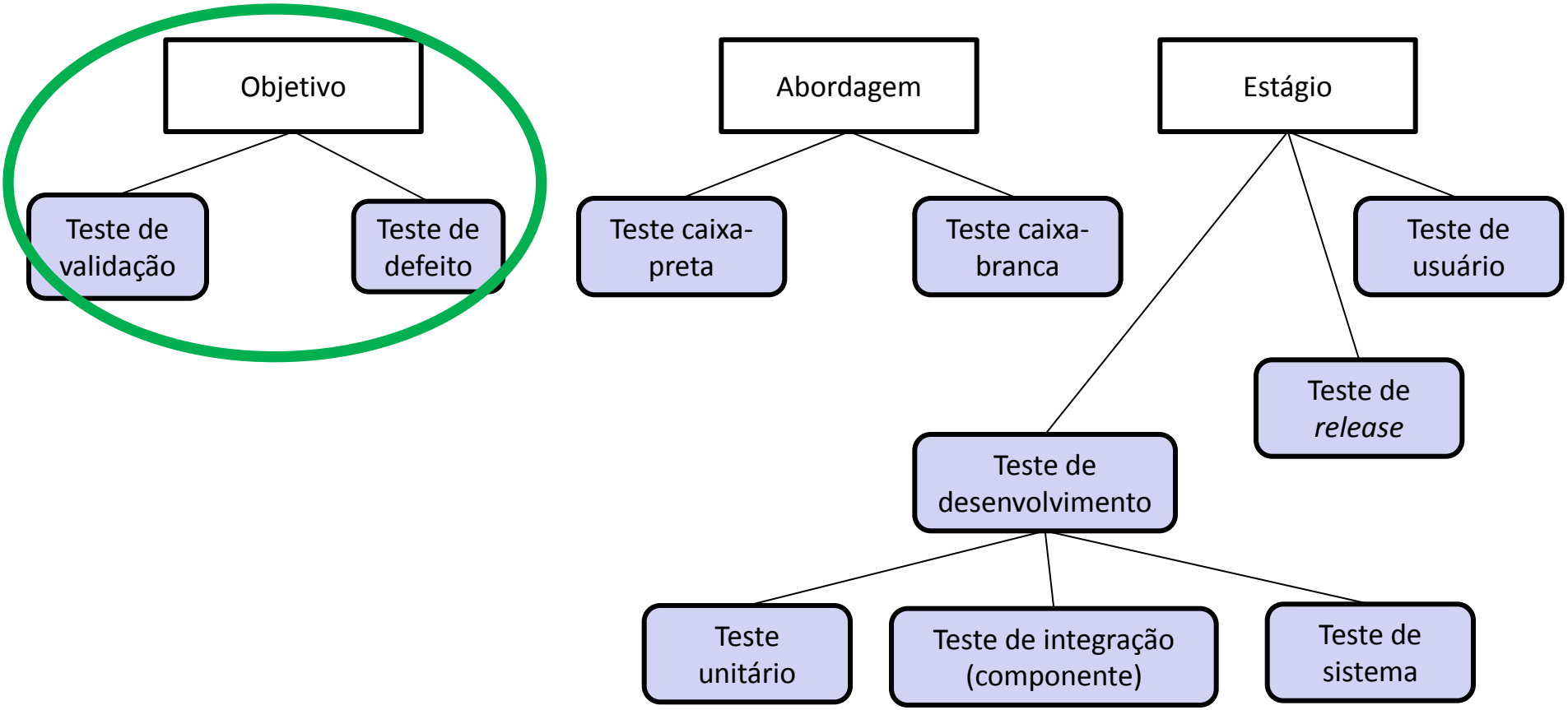
- Casos de teste
  - Declarações do que será testado
  - Especificações das entradas para o teste
  - Especificações das saídas esperadas do sistema

- Dados de teste
  - Entradas criadas para o sistema
  - Eles podem ser gerados automaticamente

Resultado e relatório

- Resultados de teste
  - Saídas que somente podem ser previstas por pessoas que conhecem o domínio de negócio do sistema
- Relatório de teste
  - Pode ser feito de forma manual, seguindo um formulário específico
  - Pode ser automatizado comparando os resultados esperados às saídas dos testes

Classificação dos testes



Classificação dos testes pelo objetivo

- Teste de validação
  - Mostrar que um programa faz o que é proposto a fazer
- Teste de defeito
  - Descobrir os defeitos do programa antes do uso

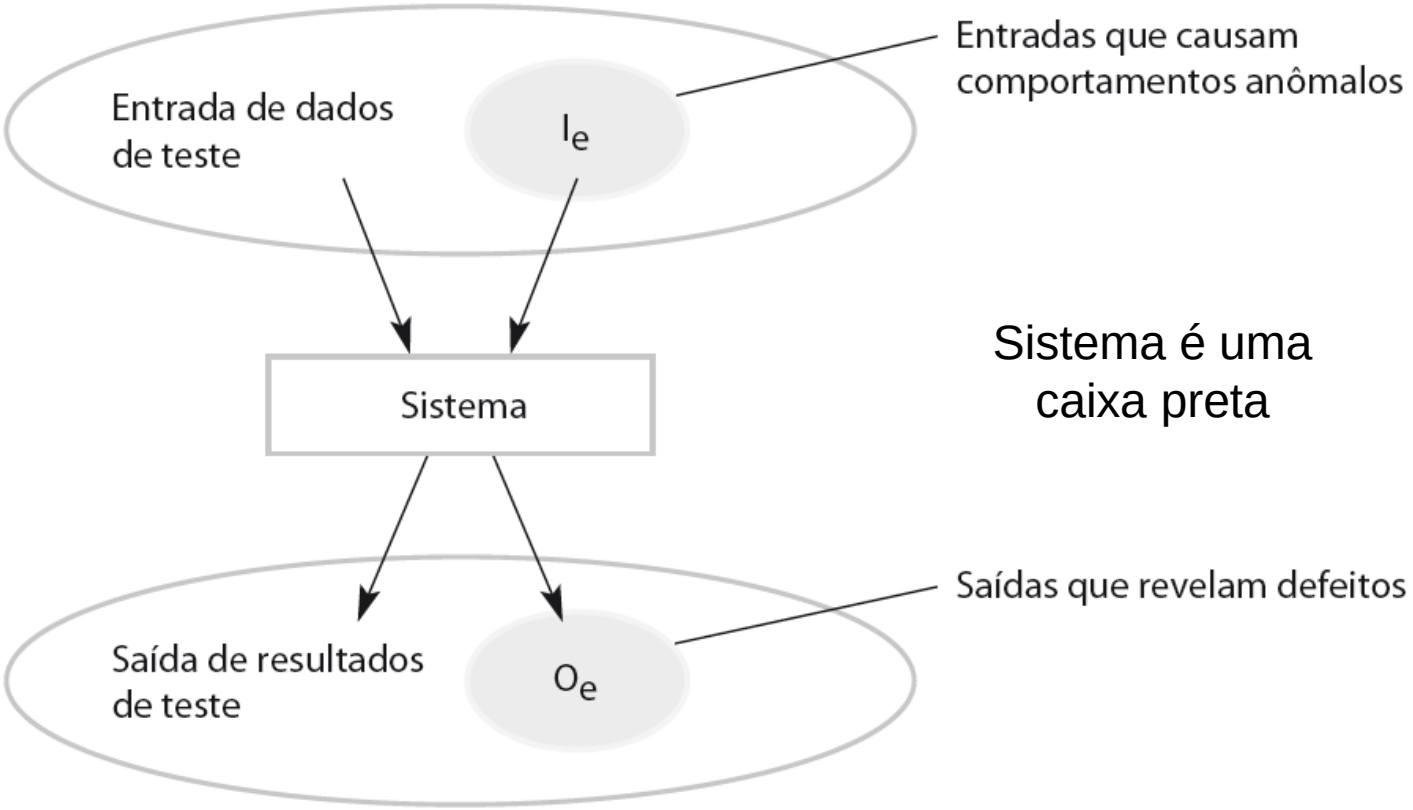
Teste de validação

- Pretende mostrar que o software atende aos seus requisitos
  - Faz o que o cliente deseja
- Um teste bem sucedido mostra que o requisito foi implementado
- Refletem o uso esperado do software

Teste de defeito

- Destinado a revelar defeitos no sistema
- Um teste de defeitos bem sucedido é aquele que revela defeitos no sistema
- Os casos de teste podem ser obscuros
  - Não precisam refletir exatamente como o sistema é normalmente usado

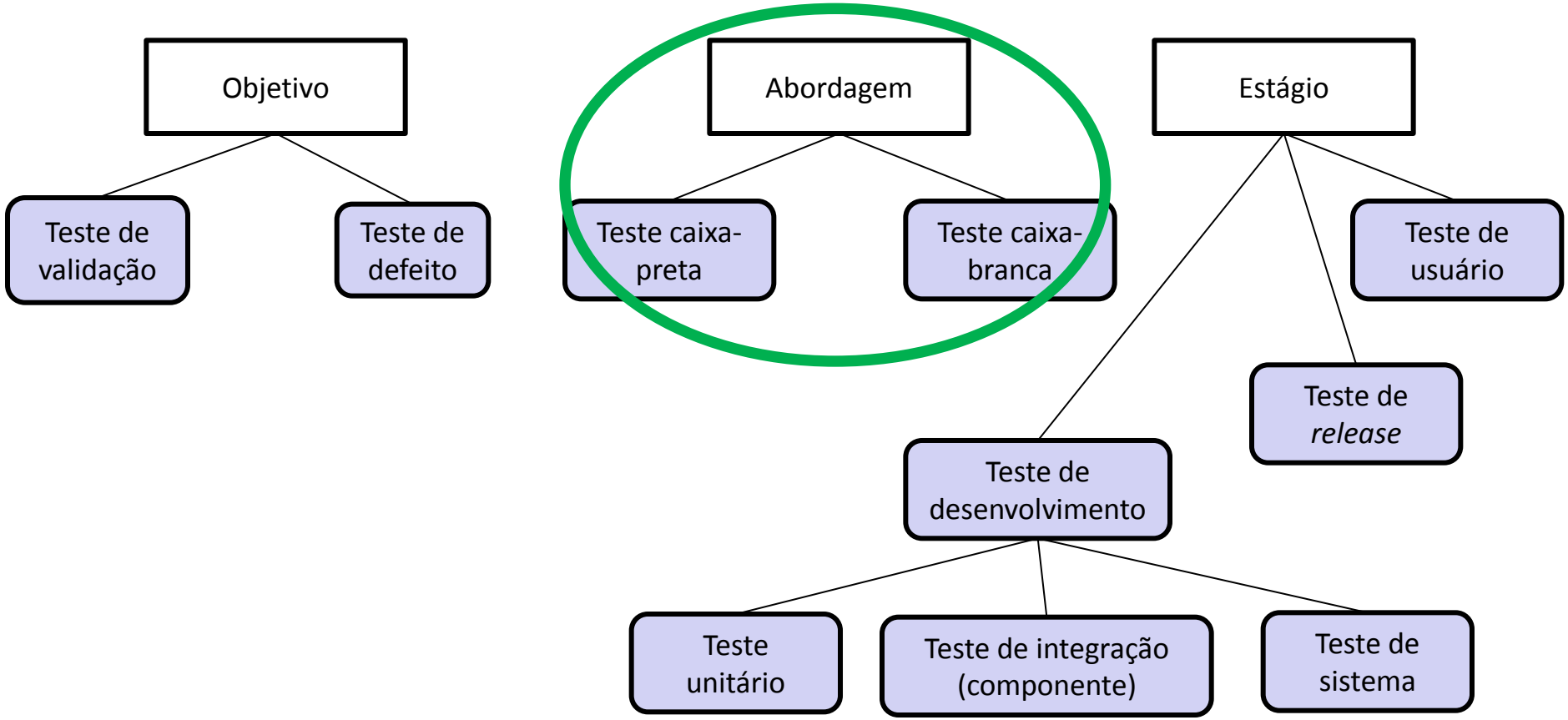
Modelo de entrada e saída



Modelo de entrada e saída

- Dado que:
  - O conjunto de entradas I gera um conjunto de saídas O
  - Algumas entradas erradas I_e geram saídas com defeitos O_e
- Testes de defeito buscam encontrar as entradas em I_e que revelam saídas em O_e
- Testes de validação envolvem entradas corretas I (não incluem entradas em I_e)

Classificação dos testes

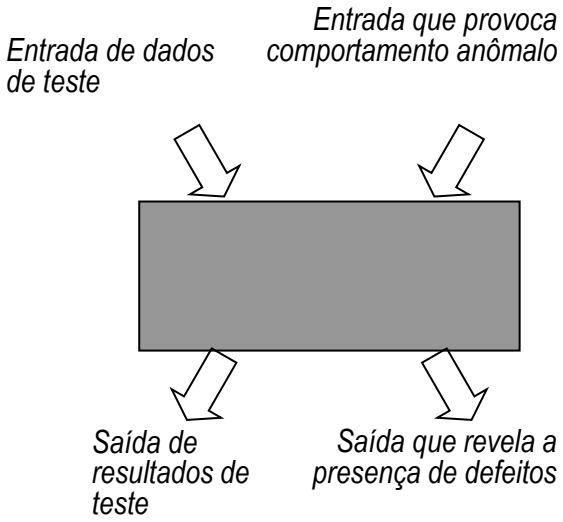


Classificação do testes pela abordagem

- Teste caixa-preta
  - Abordagem funcional
- Teste caixa-branca
  - Abordagem estrutural

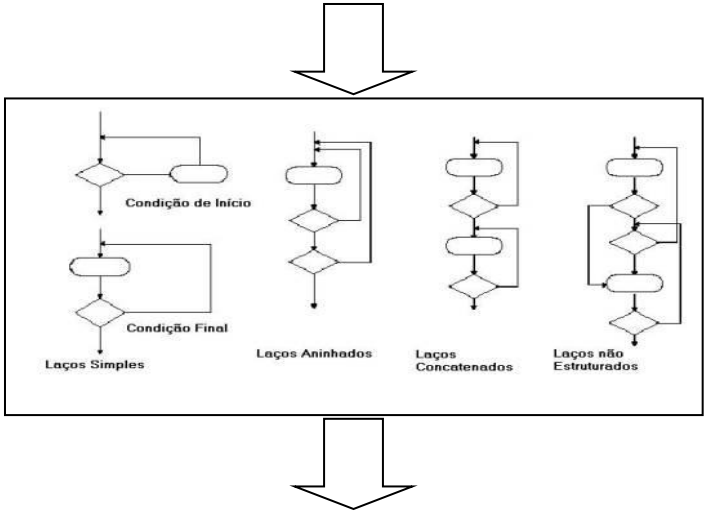
Teste caixa-preta

- Os casos de teste são gerados a partir de uma análise dos relacionamentos entre os dados de entrada e saída, com base nos requisitos levantados com os usuários
- Demonstra a operacionalidade da função
- Detalhes de implementação não são considerados



Teste caixa-branca

- Os casos de teste são gerados a partir de uma análise dos caminhos lógicos possíveis de serem executados, de modo a conhecer o funcionamento interno dos componentes
- Exercita os caminhos lógicos do programa
- Verifica se as operações internas funcionam como especificado

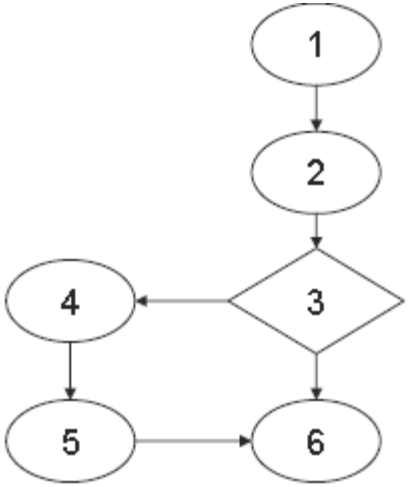


Teste caixa-branca

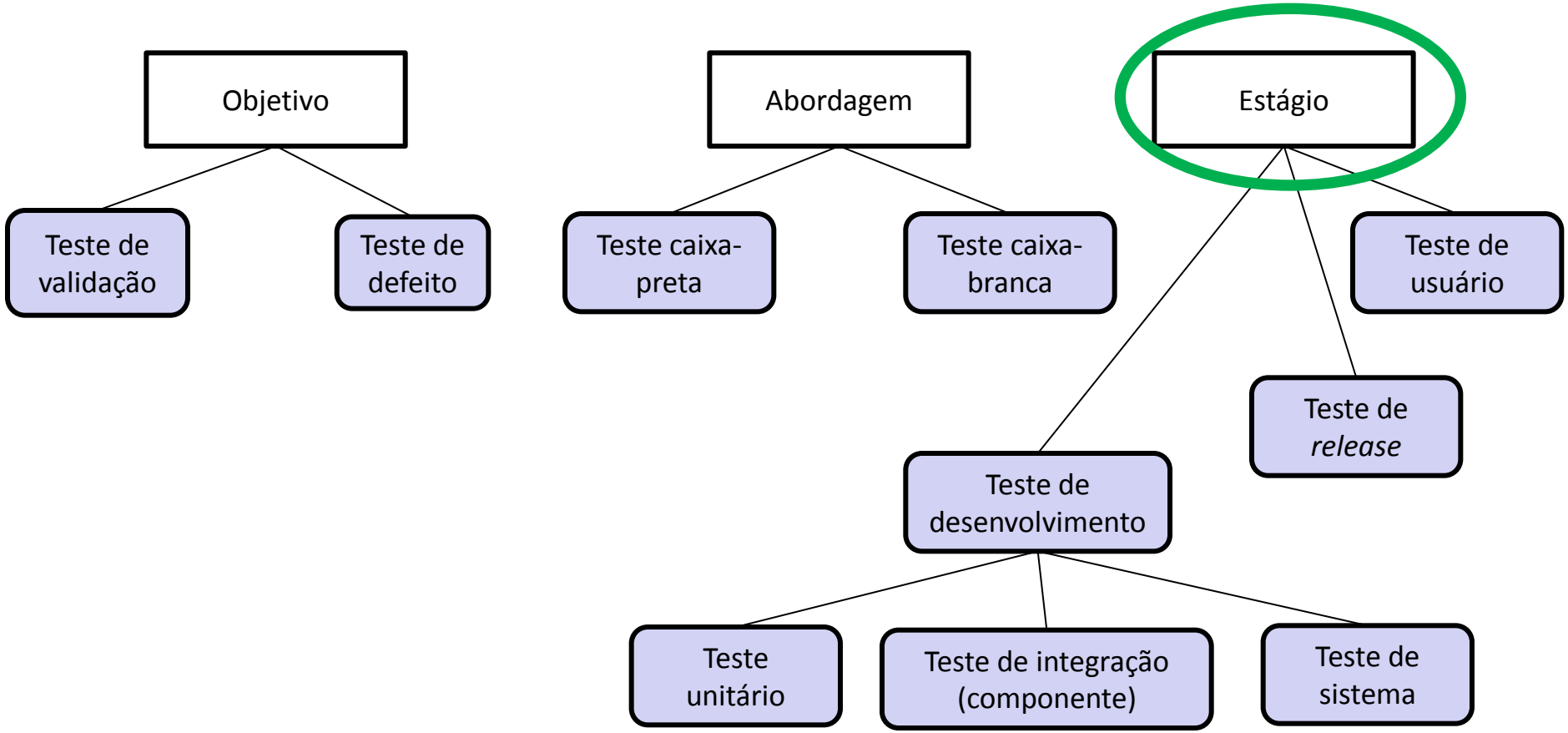
- O objetivo é assegurar que cada caminho do programa é executado pelo menos uma vez
  - Ponto de partida do teste de caminho é um fluxograma de programa

1 – 2 – 3 – 4 – 5 – 6

1 – 2 – 3 – 6



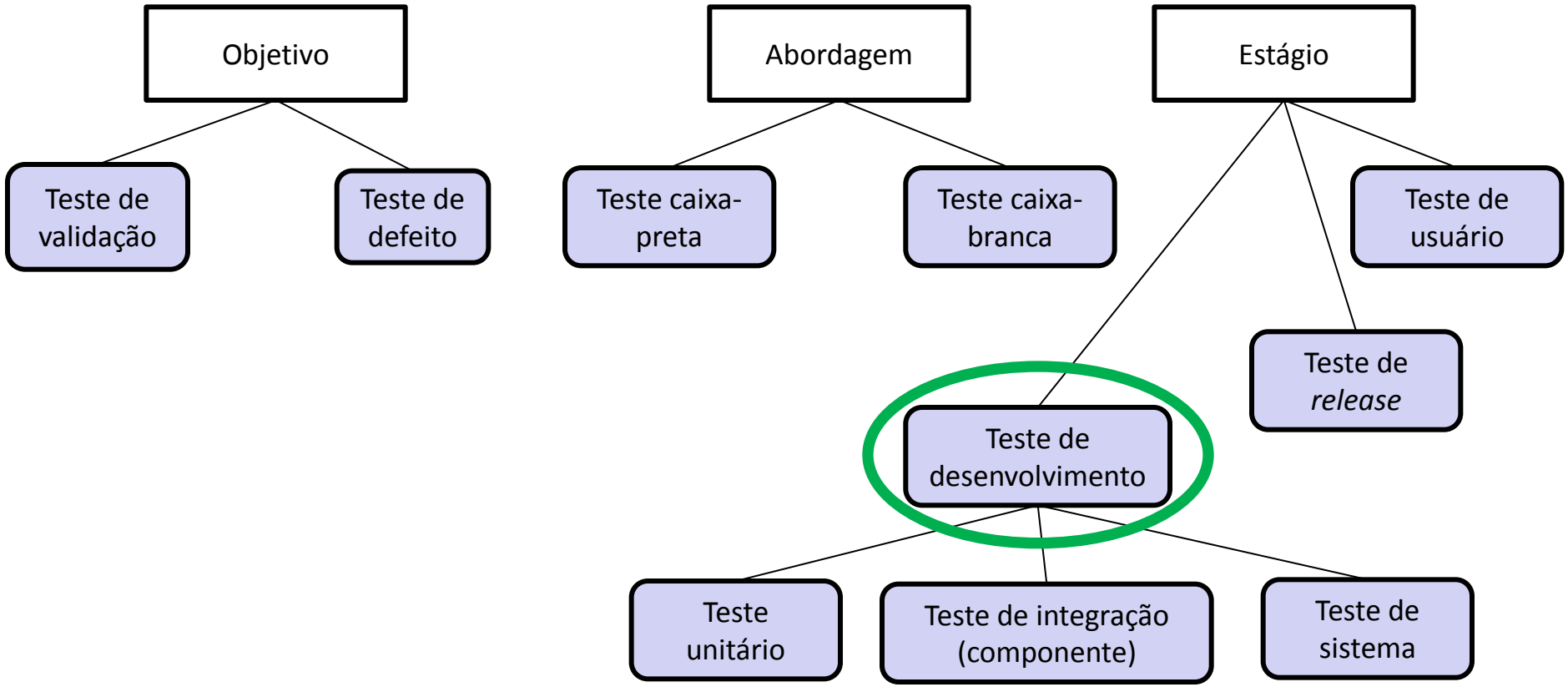
Classificação dos testes



Estágios de testes

- Testes em desenvolvimento
  - Sistema é testado durante o desenvolvimento para descobrir *bugs* e defeitos
- Testes de *release*
  - Uma equipe de teste independente testa uma versão completa do sistema antes que ele seja liberado para o usuário
- Testes de usuário
  - Usuários testam o sistema em seu próprio ambiente

Classificação dos testes



Teste de desenvolvimento

- O sistema é testado durante o desenvolvimento para descobrir defeitos
  - Os próprios programadores são responsáveis pelos testes

- Durante o desenvolvimento, o teste pode ocorrer em 3 níveis de granularidade:
  - Teste unitário
  - Teste de integração (componente)
  - Teste de sistema

Teste unitário

- Objetivo é garantir que uma unidade ou classe funciona
  - Testa unidades individuais de programa de forma independente

- Geralmente é de responsabilidade do próprio desenvolvedor da unidade
  - Os testes são derivados da experiência do desenvolvedor

Teste de classe (OO)

- O teste completo de uma classe de objetos requer
  - Teste de todas as operações associadas com um objeto
  - Atribuir e obter valores a todos os atributos de objeto
  - Exercício do objeto em todos os estados possíveis
- A herança dificulta o teste de classe

Automatização de testes

- Sempre que possível, os testes unitários devem ser automatizados

- Um teste automatizado tem 3 partes
  - Configuração: inicia o sistema com o caso de teste e dados de entrada
  - Chamada: chama o objeto a ser testado
  - Afirmação: compara o resultado da chamada ao resultado esperado

Escolha do caso de teste

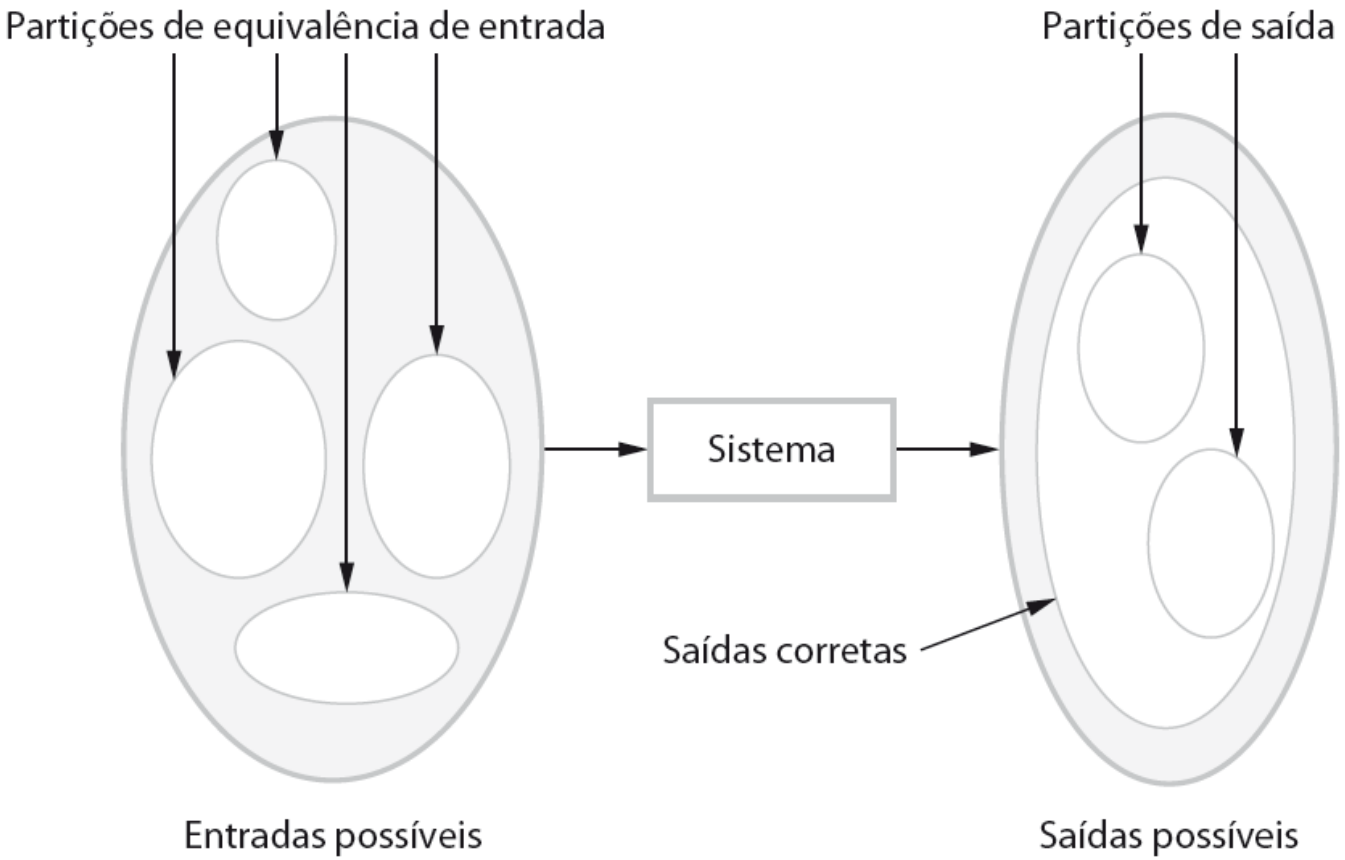
- Teste de software é caro
  - Portanto, é importante a escolha de casos de testes efetivos

- Estratégias para escolha dos testes
  - Teste de partições (caixa preta)
  - Teste estrutural (caixa branca)

Teste de partições

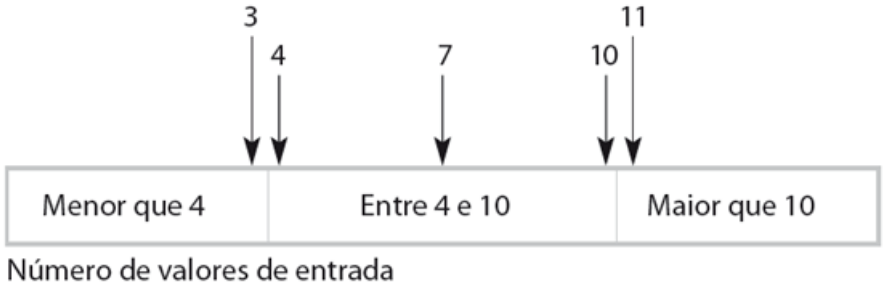
- Dados de entrada e resultados de saída podem ser particionados
  - O programa se comporta de maneira semelhante para cada partição
- Exemplos de partições
  - Números positivos / negativos
  - Itens de um mesmo menu
- Casos de teste devem ser escolhidos para exercitar cada partição

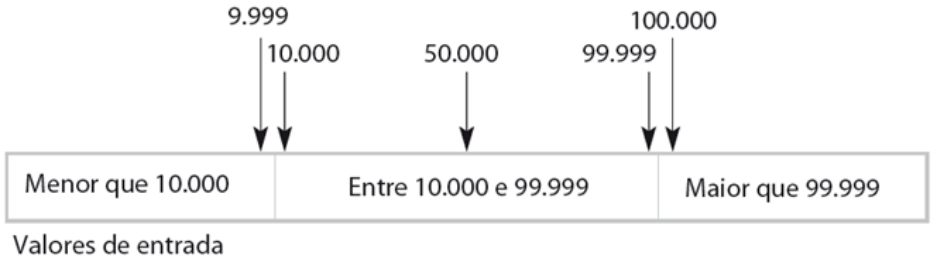
Particionamento de equivalência



Exemplos de partições

- Uma boa prática para a seleção do caso de teste é escolher casos de teste sobre os limites das partições, além de casos perto do ponto médio da partição
  - Exemplo: programa aceita entre 4 e 10 valores de entrada e cada valor é um inteiro de 5 dígitos superior a 10.000

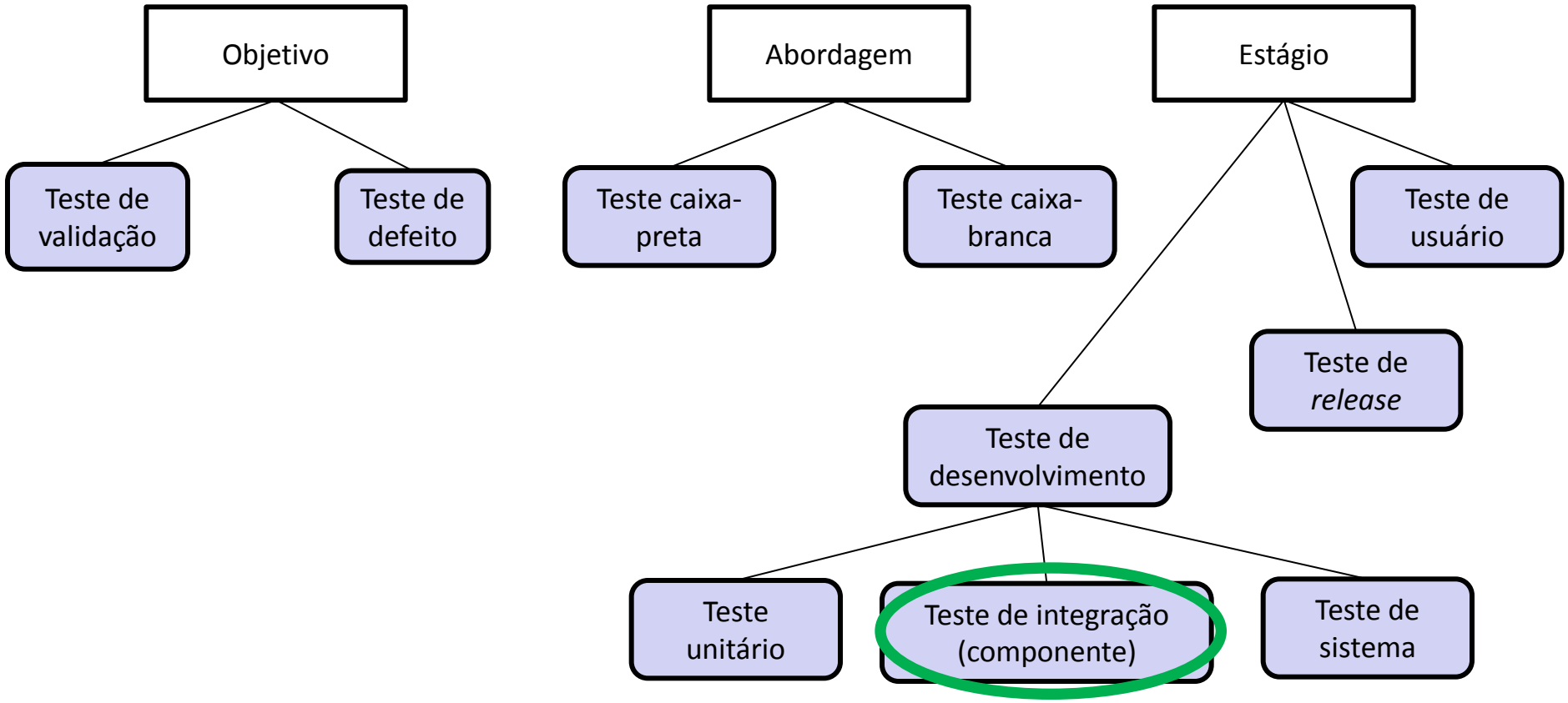




Diretrizes do teste de partições

- Testar o software com sequências de tamanhos extremos
  - Sequência de comprimento zero
  - Sequência com um único valor
  - Sequência com o tamanho máximo
- Usar sequências de tamanhos diferentes em testes diferentes
- Derivar testes para o primeiro, o médio e o último elementos da sequência

Classificação dos testes

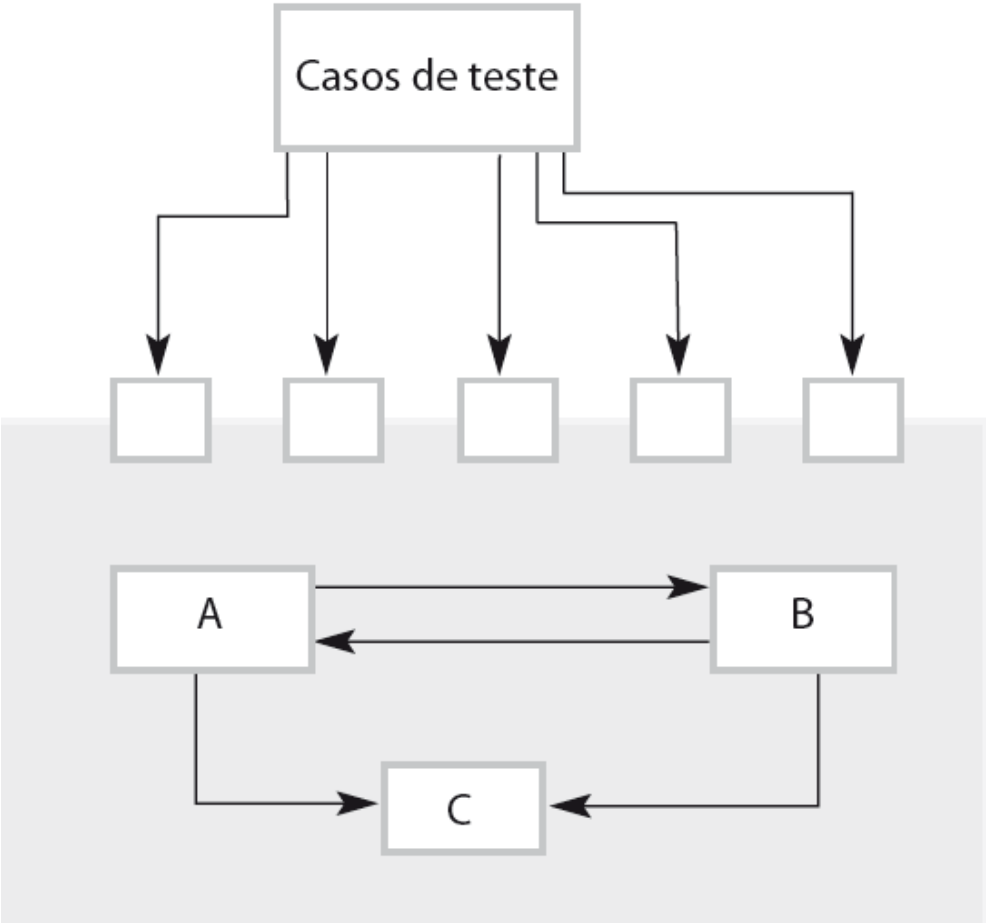


Teste de integração

- O objetivo é garantir que dois ou mais componentes funcionam juntos
  - Testa grupos de componentes integrados para criar um sistema ou um subsistema
  - Os testes se concentram nas interfaces de comunicação entre componentes

- A geralmente responsabilidade é de uma equipe independente de teste

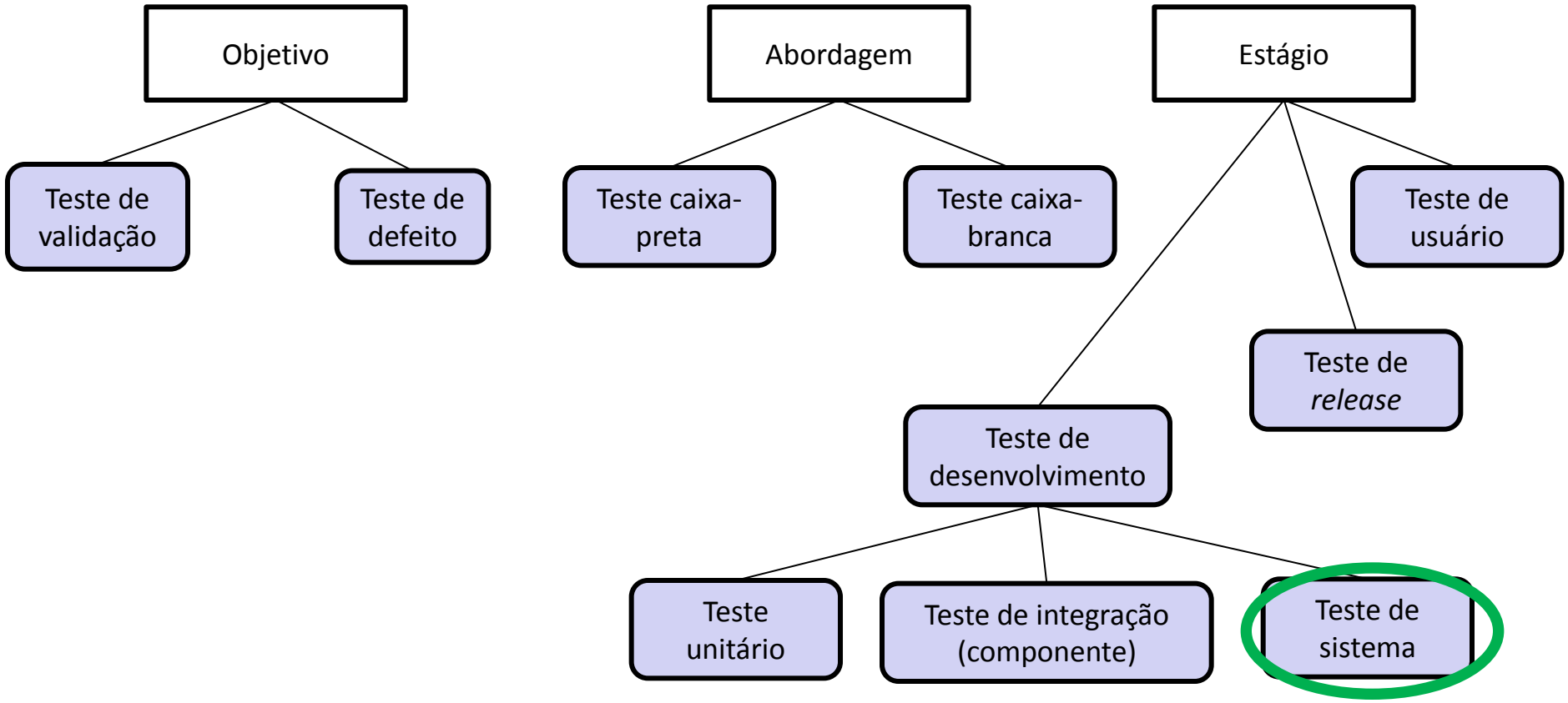
Teste de integração



Teste de integração

- Diretrizes gerais:
  - Testar valores extremos dos parâmetros para os componentes externos
  - Testar a interface com os parâmetros de ponteiros nulos
  - Testar falha nos componentes
  - Usar testes de estresse (forçam o sistema além de sua carga máxima de projeto) em sistemas de passagem de mensagem
  - Testar a variação da ordem em que componentes que interagem por meio de memória compartilhada são ativados

Classificação dos testes



Teste de sistema

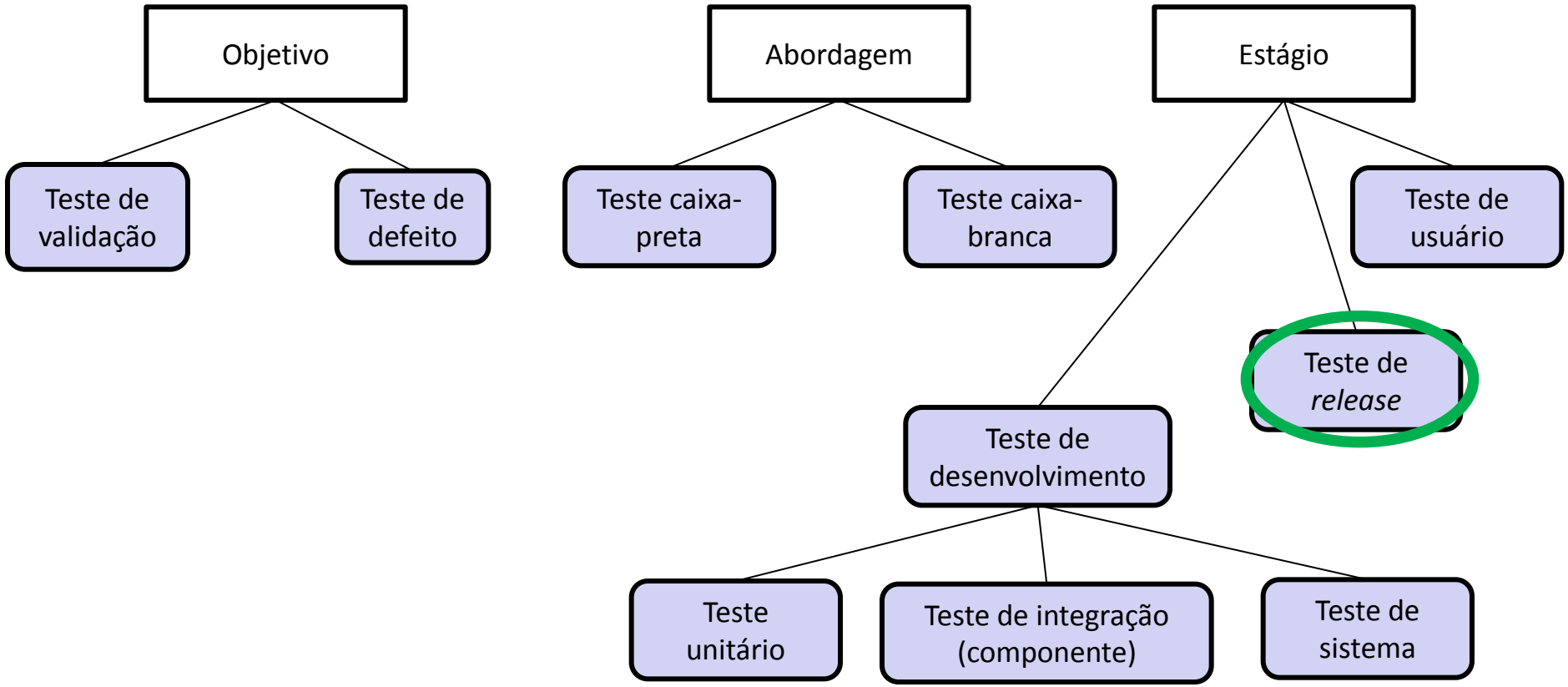
- O sistema é testado como um todo em ambiente de desenvolvimento
- Os componentes reusáveis e de prateleira são integrados com componentes recém-desenvolvidos com o objetivo de testar o sistema completo
- É um processo coletivo e não individual

Diretrizes para teste de sistema

- É impossível fazer testes exaustivos em um sistema

- Algumas diretrizes
  - Todas as funções acessadas por menus devem ser testadas
  - As combinações de funções dos mesmos menus devem ser testadas
  - As funções devem ser testadas com entradas corretas e incorretas

Classificação dos testes



Teste de *release*

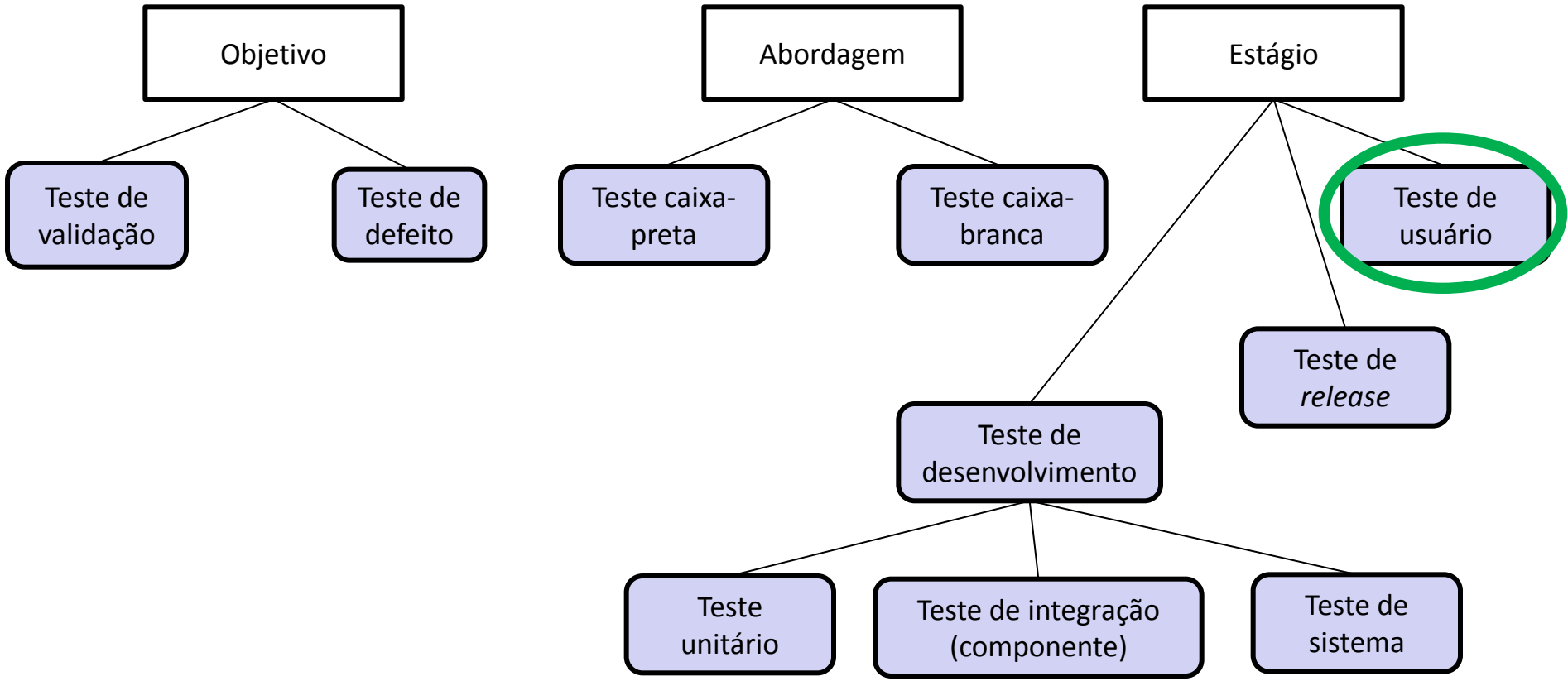
- Uma versão particular do software é testada para uso fora do ambiente de desenvolvimento

- O teste é caixa-preta
  - Nenhum conhecimento interno da estrutura do software é necessária

Teste de *release*

- Diferenças entre teste de sistema e de *release*
  - Uma equipe separada, que não esteve envolvida no desenvolvimento do sistema, deve ser responsável pelo teste de *release*
  - Testes de sistema devem centrar-se na descoberta de bugs no sistema (teste de defeito)
  - Testes de *release* devem verificar se o sistema atende a seus requisitos e é bom o suficiente para uso externo (teste de validação)

Classificação dos testes



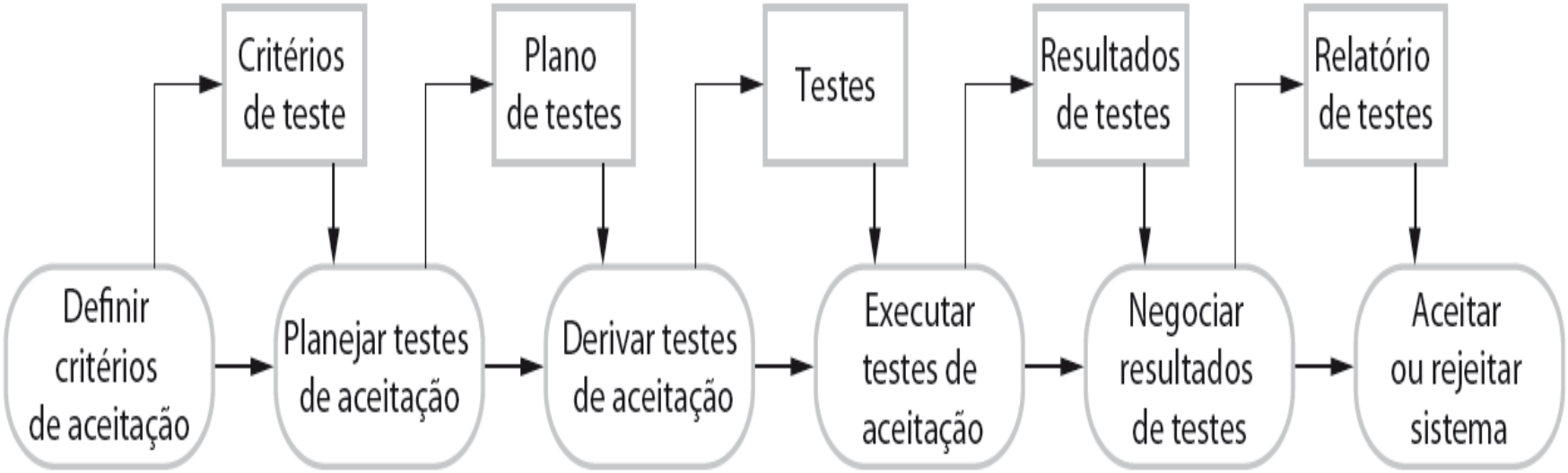
Testes de usuário

- Usuários ou clientes fornecem informações e conselhos sobre os testes de sistema
- São essenciais, mesmo quando já foram realizados os testes de sistema e de *release* abrangentes
  - As influências do ambiente de trabalho do usuário tem um efeito importante sobre a confiabilidade, desempenho, usabilidade e robustez de um sistema

Testes de usuário

- Testes alfa
  - Usuários do software trabalham com a equipe de desenvolvimento para testar o software no local do desenvolvedor
- Testes beta
  - Um release do software é disponibilizado para os usuários para que possam experimentar e levantar os problemas descobertos com os desenvolvedores do sistema
- Testes de aceitação
  - Clientes testam um sistema para decidir se se esse está pronto para ser aceito dos desenvolvedores do sistema, e implantado no ambiente do cliente

O processo de testes de aceitação



Testes: dificuldades e problemas

- Por que erros não são encontrados nos testes, apesar de investimento cuidadoso?
  - Execução de código não testado
  - Ordem de execução de sentenças diferente
  - Uso de combinação de valores fora da faixa testada
  - Ambiente de operação nunca foi testado

Testes: dificuldades e problemas

- É possível testar mais? É possível testar tudo?
  - Nem sempre é possível testar mais
  - Quase nunca é possível testar tudo

- É possível testar melhor

Testes: dificuldades e problemas

- Problema de economia (Humphrey):
  - Nunca será possível encontrar todos os defeitos em um sistema complexo
  - Nunca se provará que um sistema complexo funciona completamente apenas por testes
  - Problema: como determinar quando um sistema está bom o suficiente para encerrar a atividade de testes?
    - Normalmente quando acaba o tempo de testes
    - O retrabalho pode ser maior que alguns testes extras (até 50%)

Testes: dificuldades e problemas

- Problema social (resistência)
  - Muitos acham testar software atividade desinteressante
    - Descobrir e evitar erros são tarefas criativas
      - Exigem habilidade mental
    - Testar o código aumenta a confiança no próprio trabalho
    - Testar é aterrorizante
      - Não achamos erros no próprio código tão facilmente quanto outras pessoas

Testes: dificuldades e problemas

- Habilidade mental (Myers):
  - Imagine um programa que lê o valor de três lados de um triângulo e informa se ele é isósceles, equilátero ou escaleno. Quais seriam todas as condições a serem testadas?

Testes: dificuldades e problemas

- Resposta: no mínimo, o sistema deveria apresentar um caso de testes para cada situação abaixo:
  - Para um triângulo escaleno válido?
  - Para um triângulo equilátero válido?
  - Para um triângulo isósceles válido?
  - Para cada uma das três permutações dos dois lados válidos em triângulos isósceles?
  - Em que um lado tem comprimento zero?
  - Em que um lado tem comprimento negativo?
  - Em que a soma do comprimento de dois lados é igual ao comprimento do terceiro?
  - Para cada uma das três permutações do caso 7?
  - Em que a soma do comprimento de dois lados é menor que o comprimento do terceiro?
  - Para cada uma das três permutações do caso 7?
  - Em que todos os lados tem comprimento zero?
  - Em todos seus casos de teste está especificada a saída esperada?

Testes: dificuldades e problemas

- Problema social (papéis)
  - Muitos desenvolvedores alegam que “os testadores devem testar porque eles são muito melhores no trabalho e são pagos para isso”
    - Testar um sistema não é suficiente para descobrir erros em seus componentes
    - Cada programador deve ser responsável por testar seus componentes em um ambiente isolado
    - Erros encontrados antes do sistema entrar em testes formais podem ser consertados a um custo muito mais baixo
    - O grupo de testes não consegue desempenhar seu papel se houver erros básicos em componentes

Exercícios

2. Explique por que os testes podem detectar apenas a presença de erros, e não sua ausência
3. Você foi convidado para testar um método chamado “catWhiteSpace” em um objeto “Parágrafo”. Dentro do parágrafo, as sequências de caracteres em branco são substituídas por um único caractere em branco. Identifique partições para testar esse exemplo e derive um conjunto de testes para o método “catWhiteSpace”.

Exercícios

4. O que você entende pelo termo “teste de estresse”? Sugira como você pode estressar o teste para o sistema de informação de pacientes para cuidados com saúde mental.