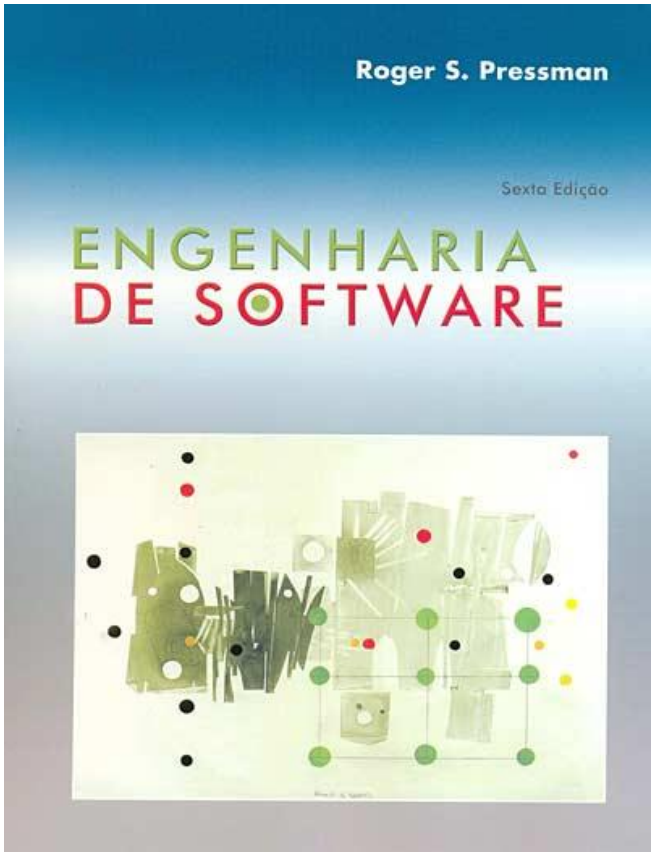
Testes de Software

Prof: Sérgio Portari



Baseado no Cap 12:
Estratégias de teste de software

Roteiro

- Conceitos de teste de software
- Atividades de teste de software
- Níveis de teste de software

Conceitos de testes de software

Conceitos de teste de software

Existe alguma diferença entre os termos defeito, falha e erro ? Ou são sinónimos ?

Depurar é uma técnica de teste de software ?

Exemplo

O fechamento não esperado de um programa.

O fechamento não esperado de um programa.

Durante um teste, ocorreu uma falha

A falha pode ser entendida como consequencia.

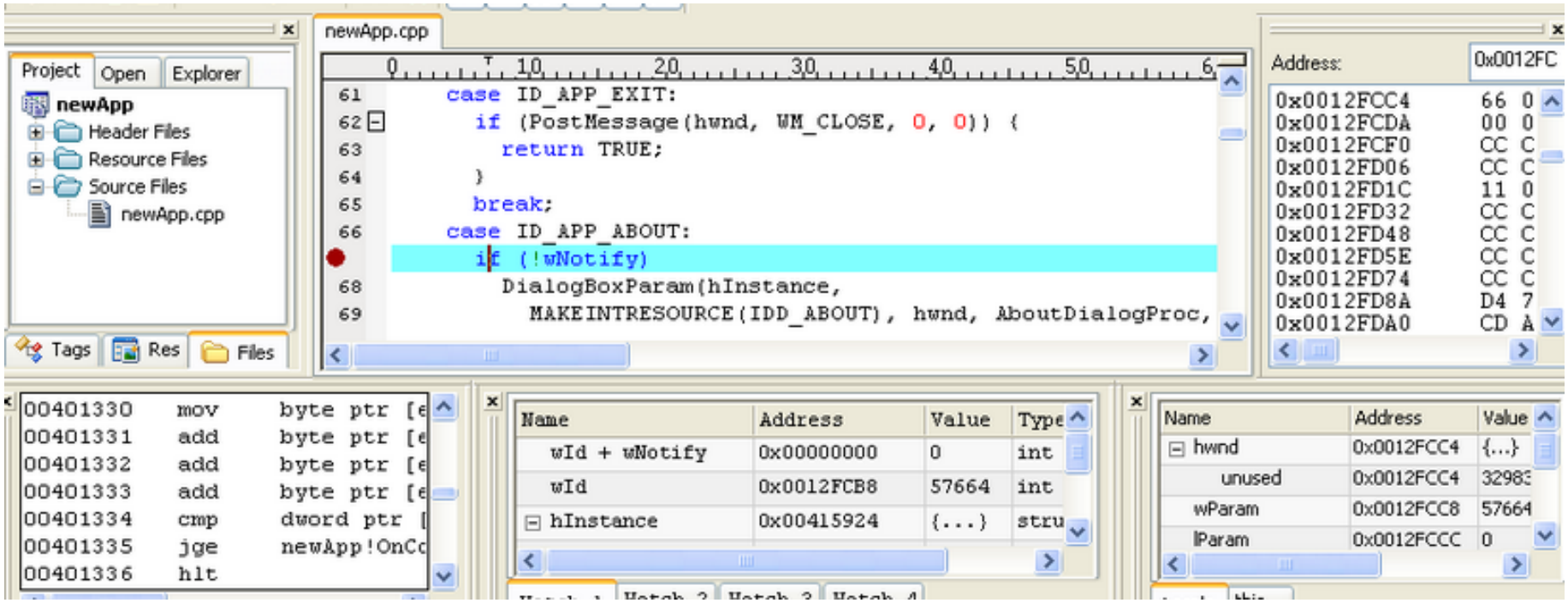
Mas qual é a causa ?

Dica:

Programa foi codificado em C++.


War3
This application has encountered a critical error:
FATAL ERROR!
Program: c:\program files\warcraft iii\war3.exe
Exception: 0xC0000005 (ACCESS_VIOLATION) at 001B:6F05EB18
The instruction at '0x6F05EB18' referenced memory at '0x1A4EF5AC'.
The memory could not be 'written'.
Press OK to terminate the application.
OK

Erro de acesso a memória



Depura-se para encontrar o defeito

Conceitos de testes de software

- A Falha foi o fechamento não esperado do software.
- O Erro foi um valor de endereço de memória inválido ou protegido.
- O Defeito, estava na codificação. Por exemplo, o programador esqueceu de inicializar uma variável.

Conceitos de testes de software

- O padrão IEEE número 610.12-1990
  - defeito (fault) – passo, processo ou definição de dados incorreto, como por exemplo, uma instrução ou comando incorreto,
  - erro (error) – diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do programa constitui um erro;
  - falha (failure) – produção de uma saída incorreta com relação á especificação.

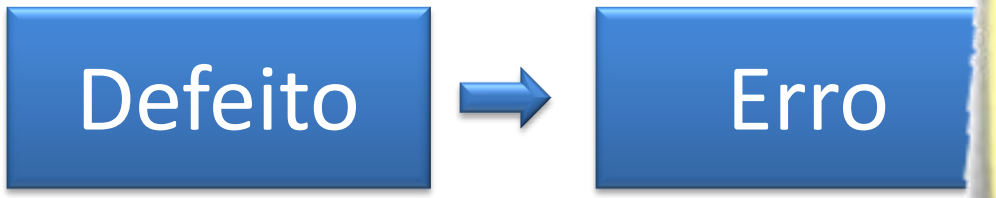
Conceitos de testes de software

Defeitos podem ocasionar a manifestação de erros.

Erros podem gerar falhas.

Falha é um comportamento inesperado que afeta diretamente o usuário.

Conceitos de testes de softwar



Uma falha pode ter sido causada por diversos erros e alguns erros podem nunca causar uma falha.

Defeitos podem ocasionar a manifestação d
erros.

Erros podem gerar falhas.

Falha é um comportamento inesperado que afeta diretamente o usuário.

Conceitos de testes de software

- Testes tem como objetivo causar falhas em um programa.

Conceitos de testes de software

- Testes tem como objetivo causar falhas em um programa.

- Depois do teste, precisamos encontrar e corrigir defeitos, ou seja, depurar.

Conceitos de testes de software

- Testes tem como objetivo causar falhas em um programa.

- Depois do teste, precisamos encontrar e corrigir defeitos, ou seja, depurar.

Depurar não é testar

Conceitos de testes de software

Testes:

Processo de execução de um programa com o objetivo de revelar a presença de falhas.
Contribuem para aumentar a confiança de que o software desempenha as funções especificadas.

Depuração:

Consequência do teste. Após revelada a presença da falha, este deve ser encontrado e corrigido.

Roteiro

- Conceitos de teste de software
- Atividades de teste de software
- Níveis de teste de software

Atividades de teste de software

Teste é um elemento de um aspecto mais amplo, referido como verificação e validação (V&V).

Atividades de teste de software

Teste é um elemento de um aspecto mais amplo, referido como verificação e validação (V&V).

Verificação e validação engloba muitas das atividades que são abrangidas para garantia de qualidade de software.

Atividades de teste de software

Verificação: Assegurar consistência, completitude e corretude do produto em cada fase e entre fases consecutivas do ciclo de vida do software.

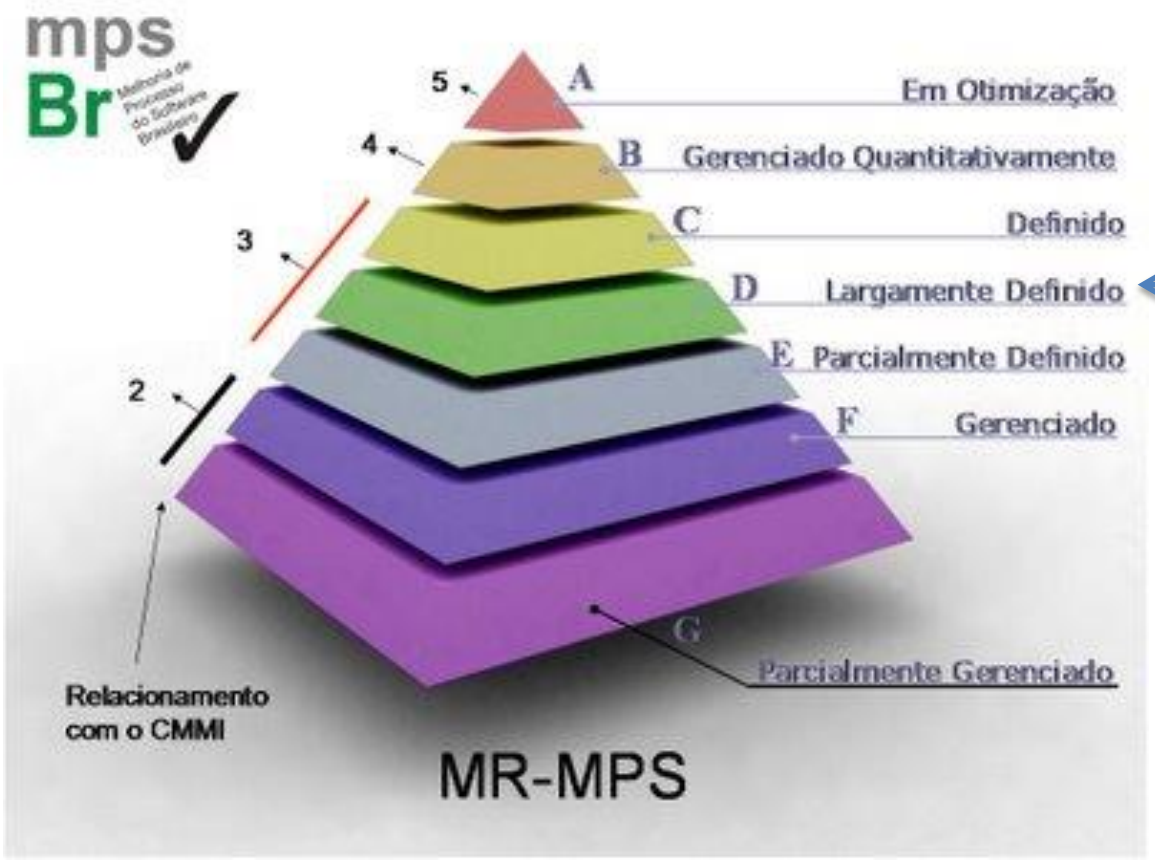
Estamos construindo corretamente o produto?

Validação: Assegurar que o produto final corresponda aos requisitos do usuário.

Estamos construindo o produto certo?

Teste: Examina o comportamento do produto por meio de sua execução.

Atividades de teste de software



Verificação e validação estão no nível D do mpsBr.

Atividades de teste de software

"Testing can only show the presence of errors, not their absence" (Dijkstra et al., 1972).

Testes podem somente mostrar a presença de erros, não a sua ausência.

Atividades de teste de software

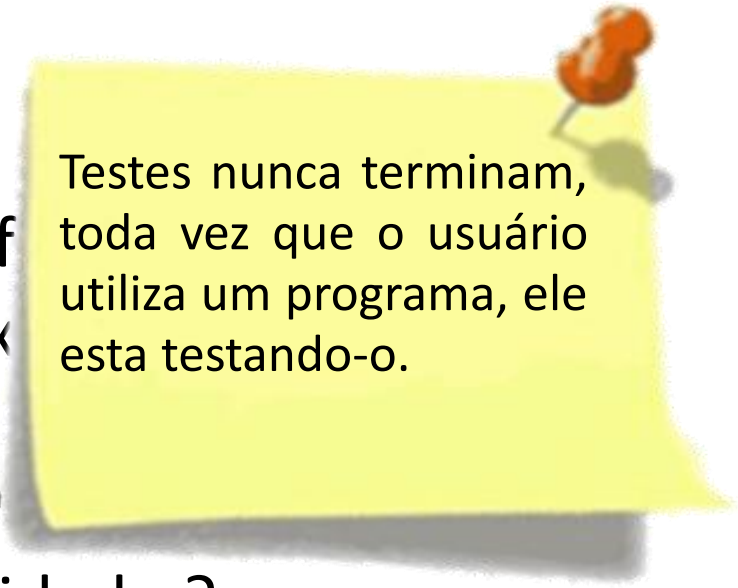
Objetivo do teste é causar uma falha, então a inexistência de falha pode ser explicado por:

- Software é de alta qualidade?
- Os testes foram de baixa qualidade ?

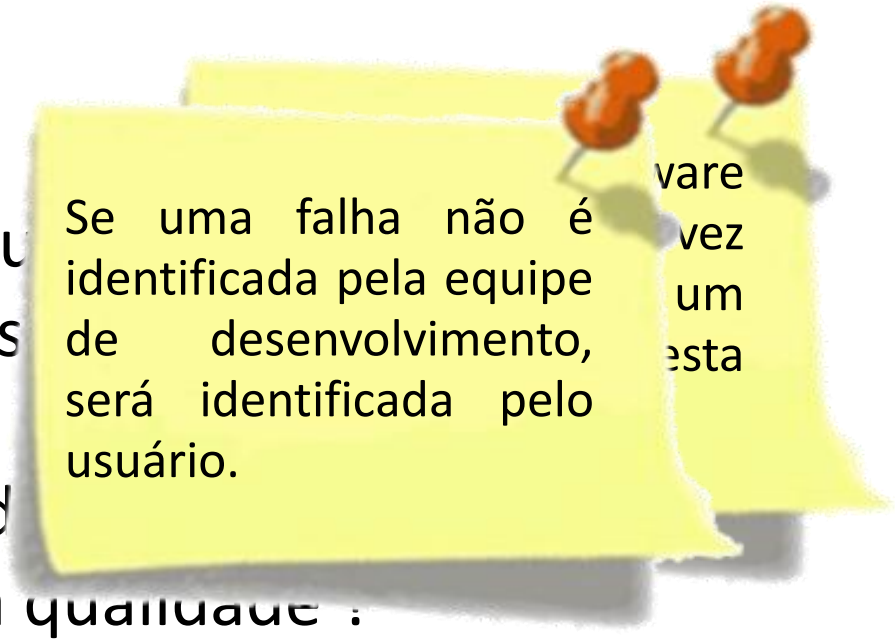
Atividades de teste de software

Objetivo do teste é causar uma f
inexistência de falha pode ser ex

- Software é de alta qualidade?
- Os testes foram de baixa qualidade ?



Atividades de teste de software



Objetivo do teste é causar u
inexistência de falha pode s

- Software é de alta qualid
- Os testes foram de baixa qualidade?

Atividades de teste de software

- Para garantir a qualidade dos testes, estes devem ser feitos de forma sistemática, que inclui:
  1. Planejamento de testes,
  2. Projeto de casos de teste,
  3. Execução e avaliação dos resultados dos testes.

Planejamento de testes

- Planejar é distribuir racionalmente no tempo os recursos disponíveis para realizar alguma atividade.
- No caso de testes será gerado o plano de teste:
  - Definir o método,
  - Recursos necessários
  - Cronograma de atividades
  - Pessoal necessário
  - O que será testado
  - O que não será testado
  - Responsáveis

Projeto de testes

- O projeto de casos de teste pode ser tão difícil quanto o projeto do próprio produto a ser testado.
- Poucos desenvolvedores gostam de teste e, menos ainda, do projeto de casos de teste
- Será escolhido um grupo específico de características a serem testadas. Descrevendo detalhadamente os métodos e testes que deverão ser executados.
- O resultado será o documento de caso de testes e o procedimento de teste.

Casos de testes

- Especificação de uma entrada para o programa e a correspondente saída esperada
  - Entrada: conjunto de dados necessários para uma execução do programa
  - Saída esperada: resultado de uma execução do programa
- Um bom caso de teste tem alta probabilidade de revelar uma falha ainda não descoberta.

Execução de teste

- Os casos de testes serão executados.
- Todo atividade de teste deve ser registrada, identificando:
  - Hora do teste
  - Procedimento
  - Pessoal envolvido
  - Resultados obtidos
  - Condições ambientais
  - Eventos não esperando (quando ocorrerem)

Roteiro

- Conceitos de teste de software
- Atividades de teste de software
- Níveis de teste de software

Quais as origens dos defeitos em software ? Onde e quando eles ocorrem ?

Defeitos no processo de desenvolvimento

Como o software é um produto lógico, defeitos são de origem humana.

Defeitos no processo de desenvolvimento

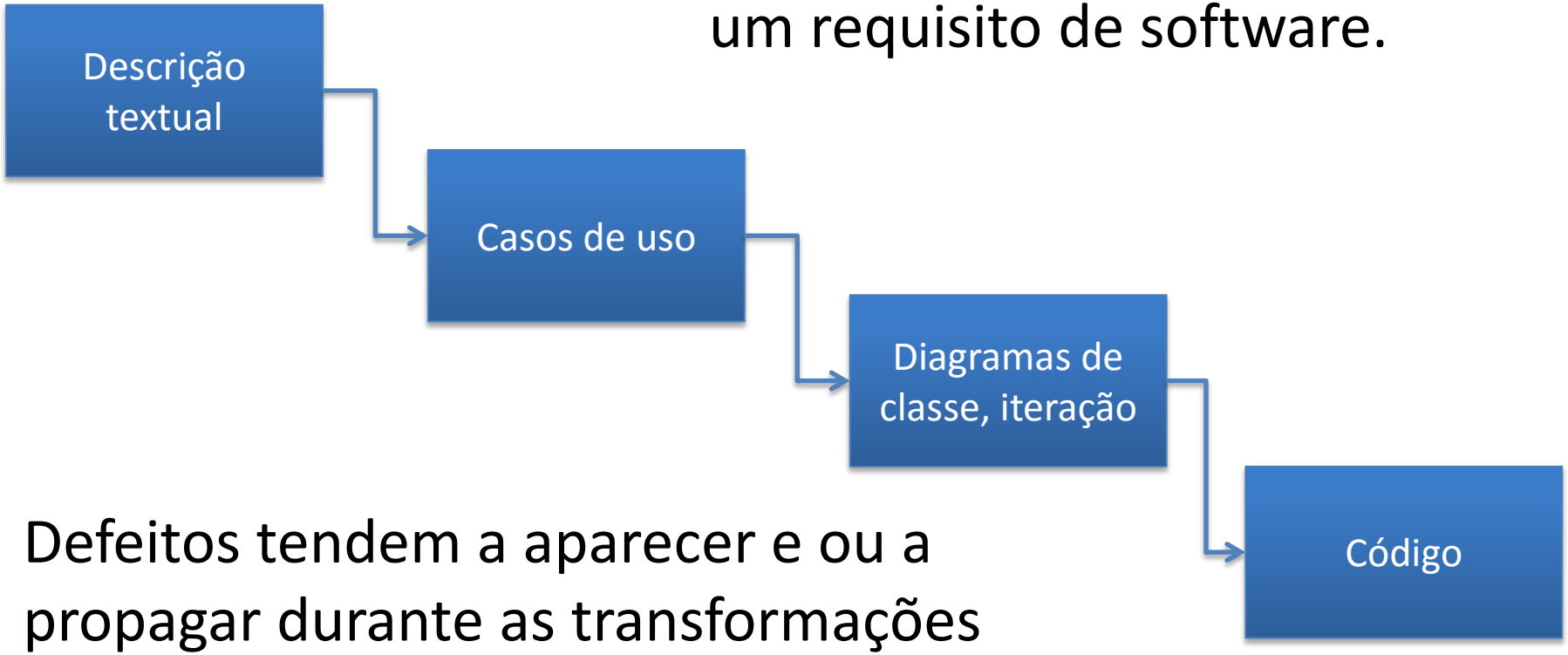
Um software falha quando ele não esta de acordo com o esperado pelo usuário. Lembrando que o esperado pelo usuário não é necessariamente o que foi especificado de modo que a falha de um sistema não esta relacionada apenas a defeitos na codificação e ou projeto.

Os defeitos normalmente são introduzidos na transformação de informações entre as diferentes fases do ciclo de desenvolvimento de um software.

Dos requisitos a codificação.

Defeitos no processo de desenvolvimento

As diversas transformações de um requisito de software.

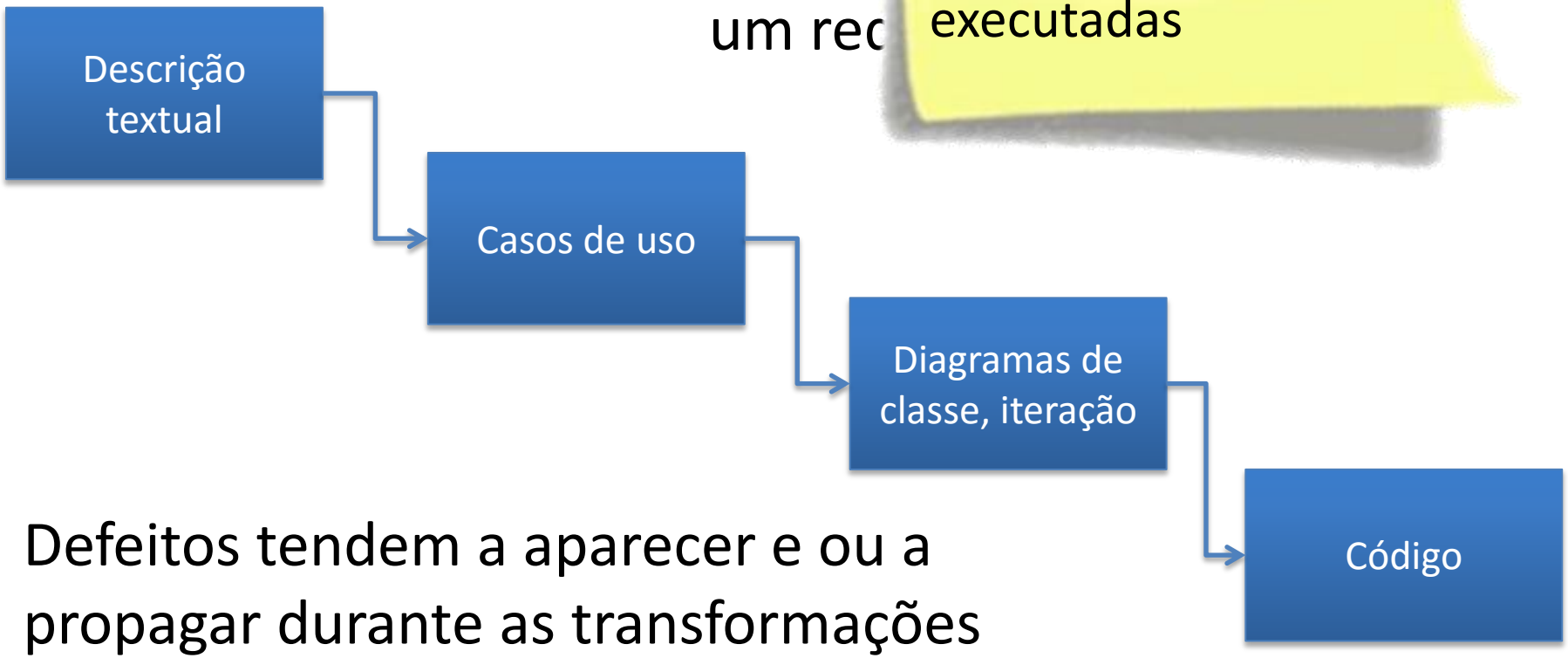


Defeitos tendem a aparecer e ou a propagar durante as transformações

Defeitos no processo de desenvolvimento

Muitos encontram-se em partes do código raramente executadas

As dive de um rec

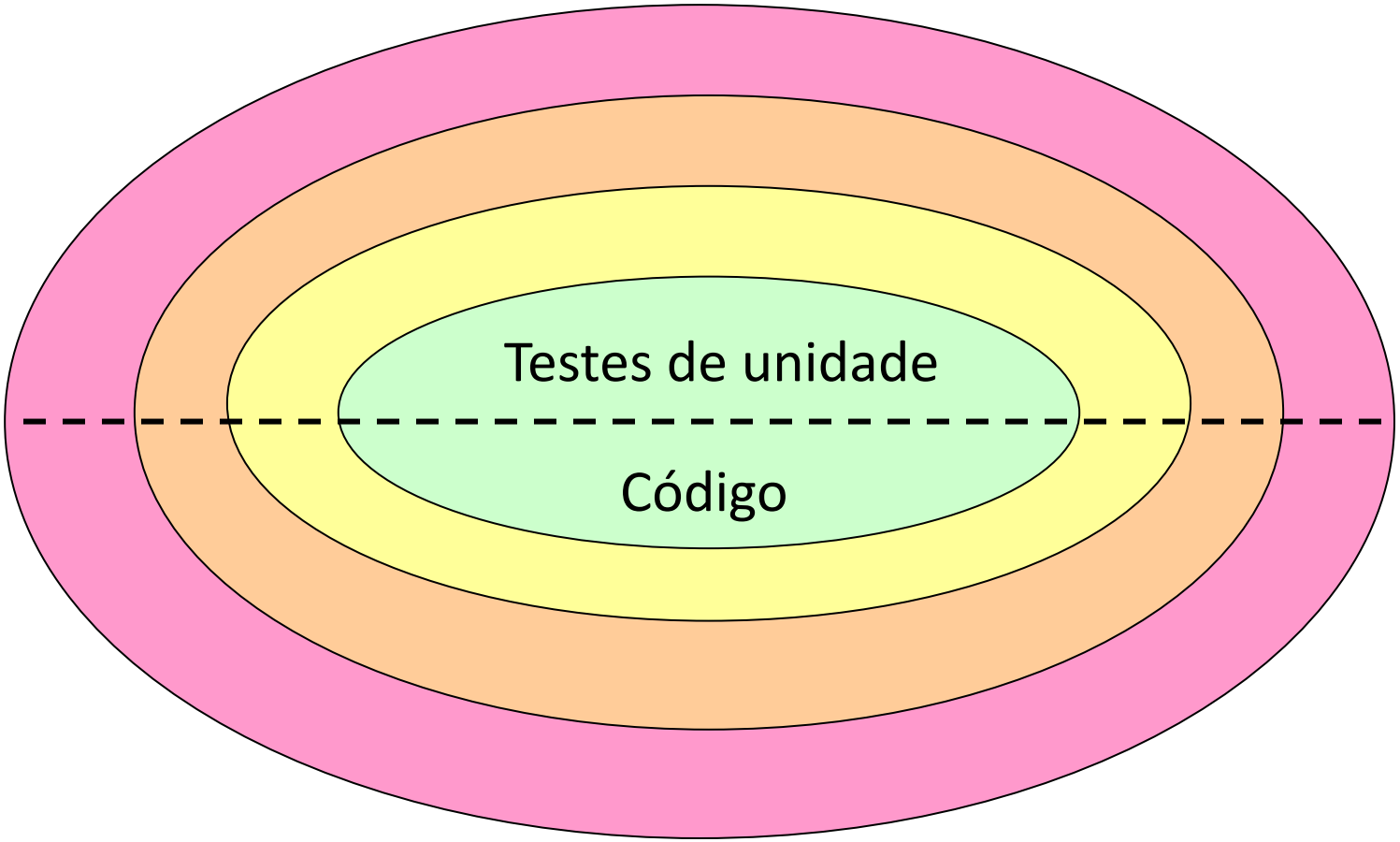


Defeitos tendem a aparecer e ou a propagar durante as transformações

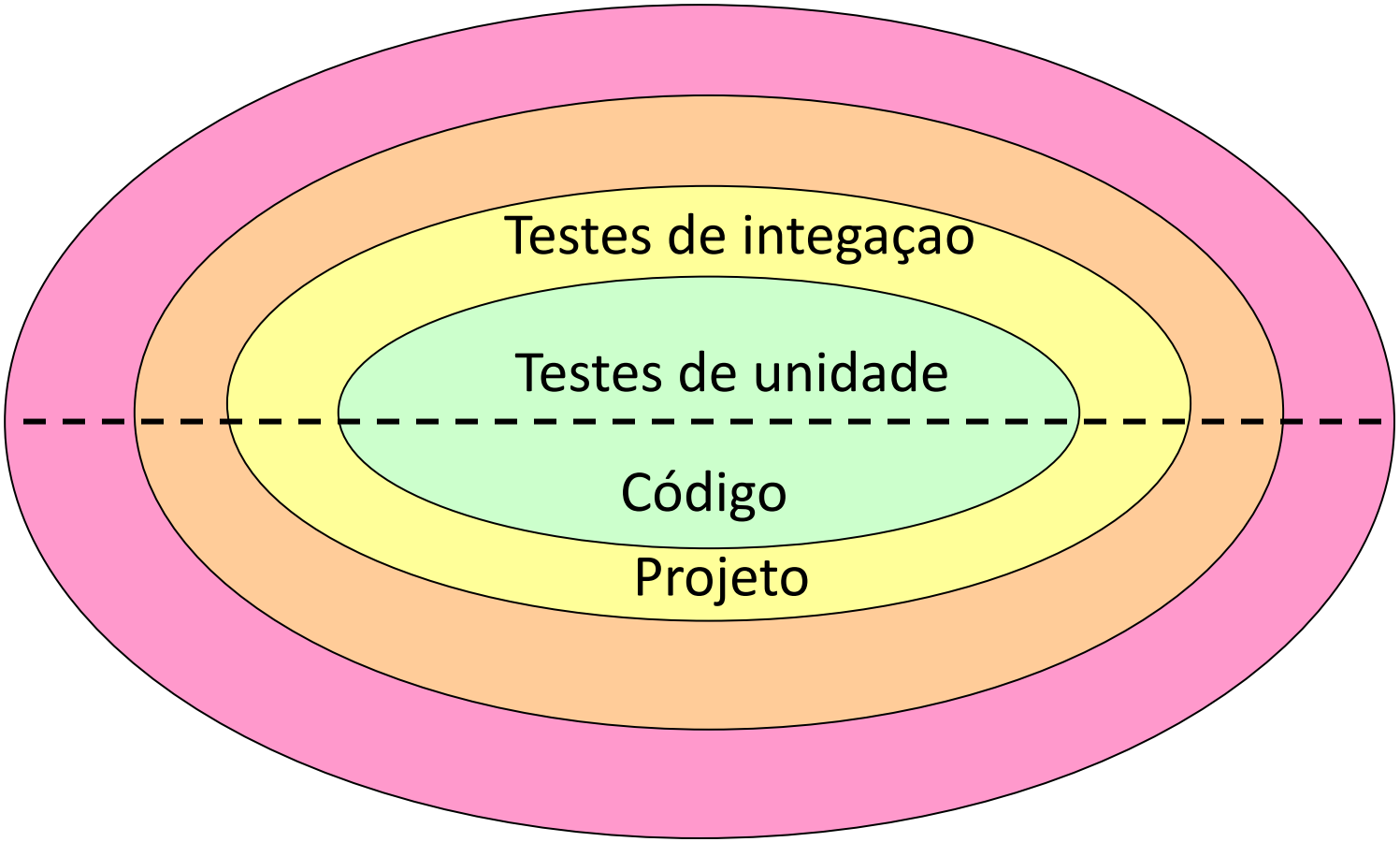
Níveis de teste de software

- Os testes devem ser executados em diferentes níveis, visando avaliar o software em diferentes perspectivas de acordo com o produto gerado em cada fase do ciclo de vida de desenvolvimento de um software (Pressman,).
  - Testes de unidade: as menores unidades funcionam corretamente?
  - Testes de integração: quando integradas elas continuam a produzir o resultado esperado ?
  - Testes de validação: (ou aceitação) o programa produz o resultado esperado pelo usuário?
  - Testes de sistema: o programa funciona como esperado no seu ambiente como todo ?

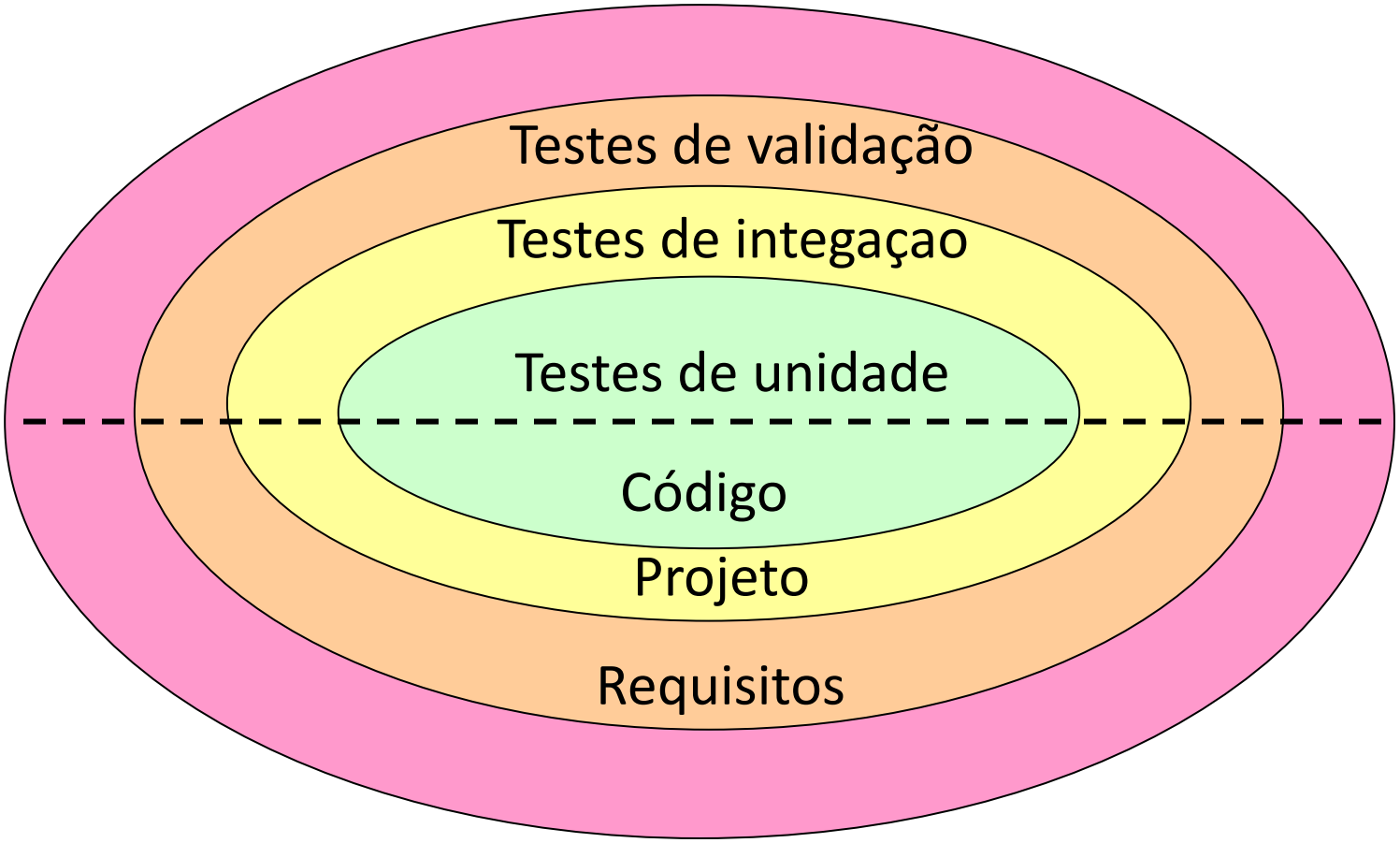
Níveis de teste de software



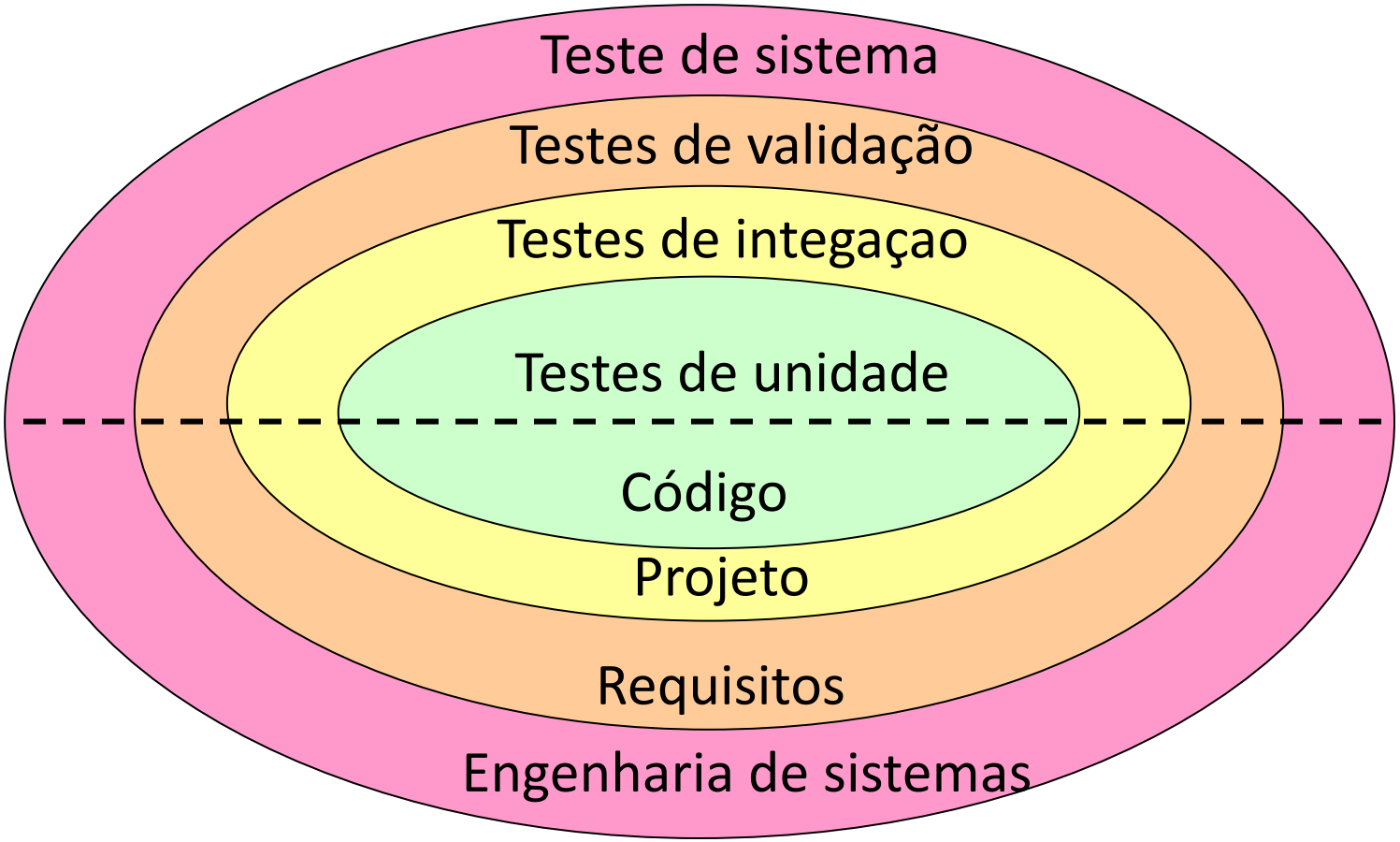
Níveis de teste de software



Níveis de teste de software



Níveis de teste de software



Níveis de teste de software

Teste de Regressão não corresponde a um nível de teste, mas é uma estratégia importante para redução de “efeitos colaterais”.

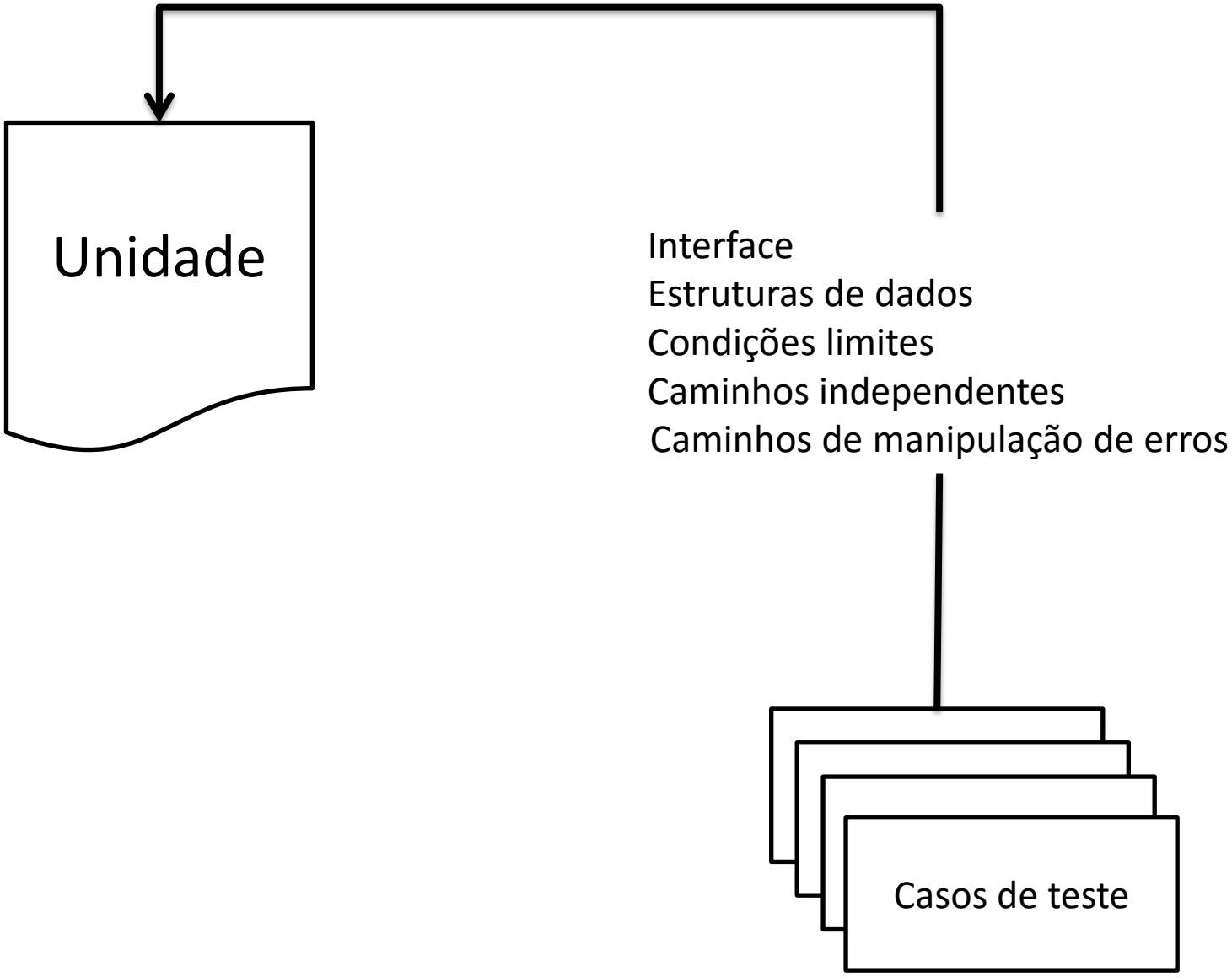
Consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema.

Pode ser aplicado em qualquer nível de teste.

Testes de unidade

- Tem como proposito testar a menor unidade do programa.
- Usa-se a descrição do projeto no nível da unidade como guia para os testes.
- Os erros estarão nos limites destas unidades.
- Pode ser conduzido em paralelo para diversas unidades.

Testes de unidade



Testes de unidade

- Interface, é testada para garantir que a informação flui adequadamente para dentro e para fora da unidade do programa.
- Estrutura de dados, é examinada para garantir que os dados armazenados temporariamente mantenham sua integridade durante todos os passos do programa.
- As condições-limites são testadas para garantir que a unidade opere adequadamente nos limiares.
- Todos os caminhos independentes (básicos) são exercitados para garantir que que todos os comandos tenham sido executados
- Todos os caminhos de manipulação de erros são testados.

Testes de unidade

- Interface, é testada para garantir que a informação flui adequadamente para dentro e para fora da unidade do programa.
- Estrutura de dados, é examinada para garantir que os dados armazenados temporariamente mantenham sua integridade durante todos os passos do programa.
- As condições-limites são testadas para garantir que a unidade opere adequadamente nos limiares.
- Todos os caminhos independentes (básicos) são exercitados para garantir que que todos os comandos tenham sido executados
- Todos os caminhos de manipulação de erros são testados.

Testes de unidade: Interface

- Testes de fluxo de dados entre as interfaces são necessários antes de qualquer outro teste.

- Se os dados não entram e nem saem adequadamente, todo os testes são discutíveis.

Testes de unidade: ED

- As estruturas de dados (ED) locais devem ser exercitadas

- O impacto local nos dados globais devem ser verificados (se possível) durante o teste de unidade.

Teste de unidade: teste de caminho

- Casos de testes devem ser projetados para descobrir erros devidos a cálculos errados, comparações incorretas ou fluxo de controle inadequado.
- Exemplos:
  - Inicialização incorreta, representação incorreta de uma expressão, erros de operadores ou precedência, variáveis de ciclo inadequadamente modificada e etc

Teste de unidade: limites

- Teste nos limites é uma das mais importantes tarefas do teste de unidade.
- O software frequentemente falha nos limites:
  - N-esimo elemento de um vetor de dimensão N é processado, a I-esima repetição de um ciclo com I passagens, valor máximo ou mínimo é encontrado e etc.

Teste de unidade: exceção

- Erros potenciais no tratamento de erro (ou exceção):
  1. Descrição de erro ininteligível
  2. Erro mencionado não corresponde ao erro encontrado
  3. A condição de erro provoca a intervenção do sistema antes da manipulação de erro
  4. O processamento da condição de exceção de erro esta incorreto.
  5. A descrição de erro não fornece informação suficiente para encontrar o defeito

Testes de unidade: procedimento

- Normalmente considerado um apêndice ao passo de codificação.
- O projeto de teste pode ser realizado antes que o código seja iniciado (abordagem ágil)
- Cada cada caso de teste deve ser acoplado a um conjunto de resultados esperados.

São capazes de identificar um desafio prático com relação aos testes de unidade?

Testes de unidade: procedimento

- Uma unidade não é um programa isolado, o software para um pseudocontrolador (driver) e ou pseudocontrolado (stub) precisa ser desenvolvido para cada caso de teste.
  - Em muitos casos o pseudocontrolador será apenas um programa principal, que aceita dados do caso de teste e passa para unidade.
  - Pseudocontrolado substitui unidades subordinados a unidade a ser testada.

Testes de unidade: procedimento

- Uma unidade não é um pr
  software para um pseudoc
  e ou pseudocontrolado (st
  desenvolvido para cada caso de teste.
  - Em muitos casos o pseudocontrolador será apenas um programa principal, que aceita dados do caso de teste e passa para unidade.
  - Pseudocontrolado substitui unidades subordinados a unidade a ser testada.

Pseudocontroladores e pseudocontrolados são despesas indiretas.

Testes de unidade: procedimento

- Uma unidade não é ur
  software para um pseu
  e ou pseudocontrolad
  desenvolvido para cad
  - Em muitos casos o pseudocontrolador será apenas um programa principal, que aceita dados do caso de teste e passa para unidade.
  - Pseudocontrolado substitui unidades subordinados a unidade a ser testada.

Pseudocontrola es e são

Em alguns casos pode não valer a pena desenvolvê-los, tendo que adiar os testes.

Como a alta coesão de uma unidade pode afetar o teste de unidade?

Testes de unidade: procedimento

- Testes de unidades são simplificados em componentes com alta coesão.
- Quando uma única função é implementada o número de casos de testes é reduzido e os erros podem ser mais facilmente previstos e descobertos.

Testes de unidade: procedimento

- Testes de unidades s
componentes com a
- Quando uma única f
número de casos de
erros podem ser ma
descobertos.

Por isso o teste de programas escritos em linguagens funcionais como Haskell tendem a ser mais fáceis. A disciplina destas linguagens torna as funções altamente coesas.

Testes de unidade

- Os testes de unidade em linguagens funcionais, como Haskell, focam nas funções.
- Em linguagens procedurais como C, os testes de unidades são em procedimentos.

Testes de unidades

Em linguagens orientadas a objetos, os testes de unidades podem focar apenas nos métodos isoladamente ?

Testes de unidade em OO

- Considerem uma hierarquia de classe, na qual uma operação X é definida para uma superclasse e herdada por várias subclasses.
- A operação X será aplicada sobre diferentes contextos.
- Precisamos testar a operação X como parte de uma classe.

Se minhas unidades funcionam como o esperado, não significa que meu software irá funcionar como o esperado ?

Testes de integração

- Infelizmente, software é um sistema complexo. Defeitos de projetos podem:
  - Perder dados entre as interfaces (ex: dados incompátiveis)
  - Efeito imprevisto ou adverso sobre o outro (ex: variáveis globais)
  - Subfunções quando integradas podem não produzir o resultado esperado pela função principal.
  - etc

Testes de integração

- Infelizmente, soft
  complexo. Defeit
  - Perder dados en
    incompátiveis)
  - Efeito impreviste
    variáveis globais
  - Subfunções quando integradas podem não produzir a função principal.
  - etc

A imutabilidade das variáveis em linguagem funcional tende a evitar algumas das falhas relacionadas a integração.

Testes de integração

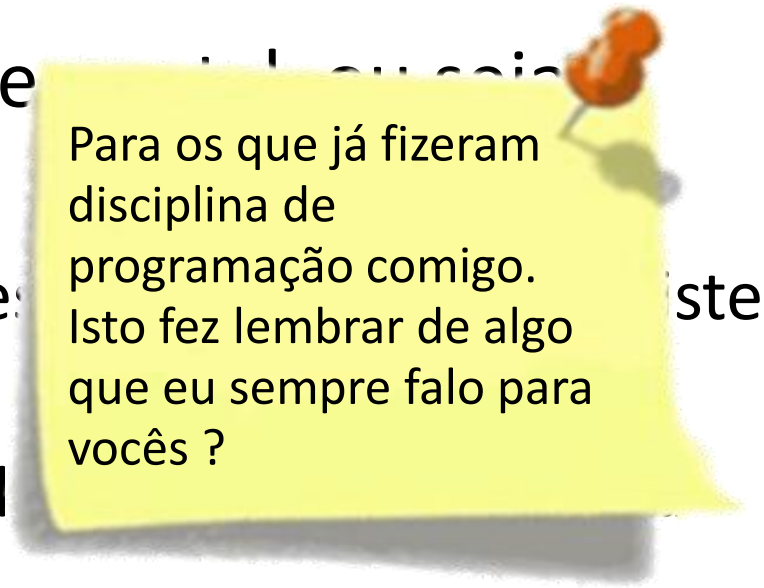
- Técnica sistemática para construir a arquitetura do software e ao mesmo tempo, conduz testes para descobrir falhas associadas a interface.
- Abordagens:
  - a) Não incremental e b) incremental

Testes de integração: não incremental

- A integração não incremental, ou seja, a abordagem big-bang.
  - Todos os componentes são integrados e o sistema é testado.
  - Não existe uma abordagem sistemática para integração
- Abordagem caótica, um conjunto de falhas é identificada e a correção é difícil, pois o isolamento das causas é complicado.

Testes de integração: não incremental

- A integração não incre… ou seja, abordagem big-bang.
  - Todos os componente… istema é testado.
  - Não existe uma abord… integração
- Abordagem caótica, um conjunto de falhas é identificada e a correção é difícil, pois o isolamento das causas é complicado.

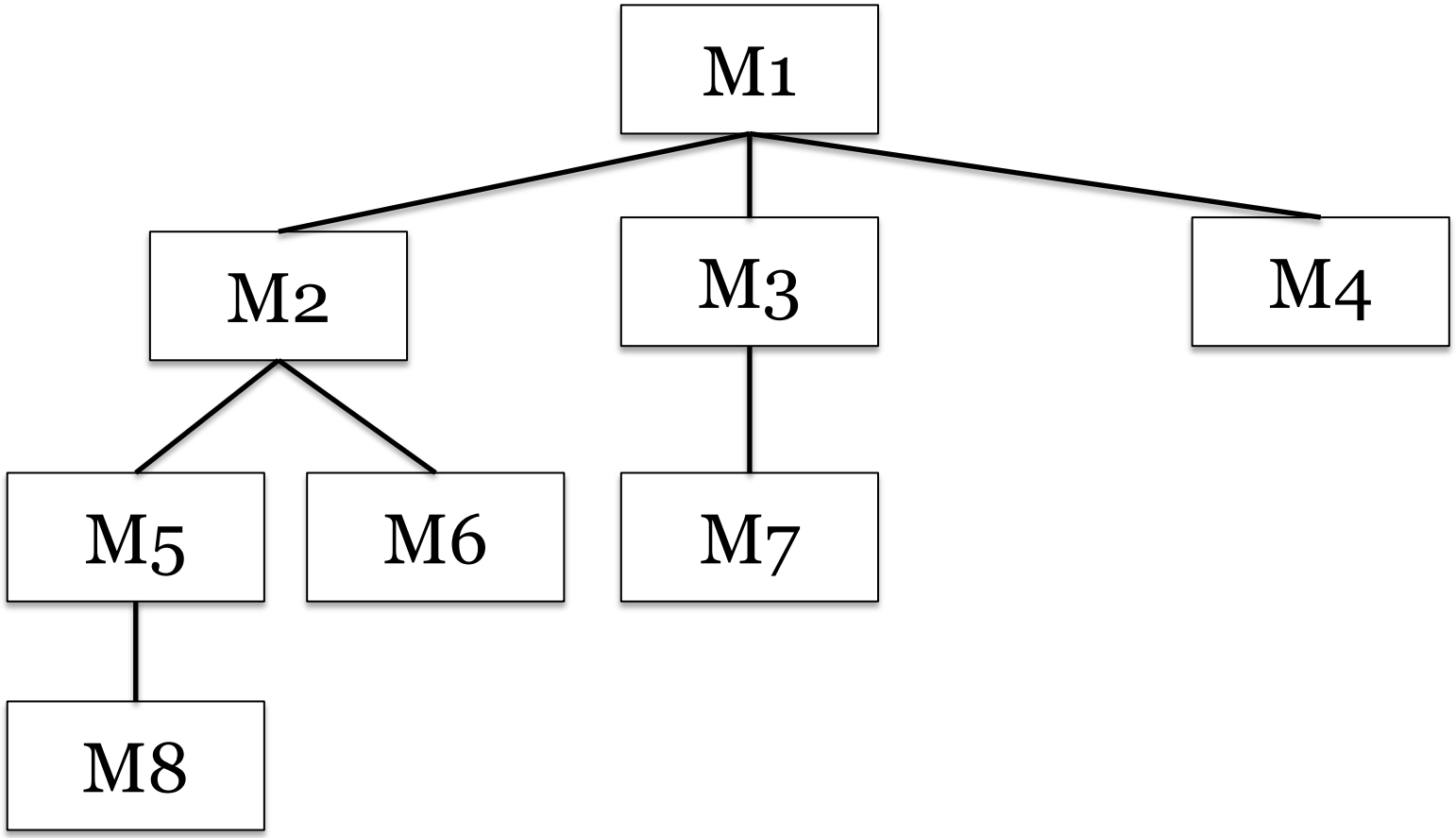


Testes de integração: incremental

- Antítese da abordagem big-bang. O programa é construído e testado em pequenos incrementos, em que os erros são mais fáceis de isolar e corrigir:
  - Integração descendente (top-down), as unidades são integradas movendo descendentemente pela hierarquia, começa pela unidade principal(programa principal) e dela as unidades subordinadas.
  - Integração ascendente (bottom-up), inicia o teste com as unidades atômicas (níveis mais baixos do programa) e são integrados de baixo para cima.

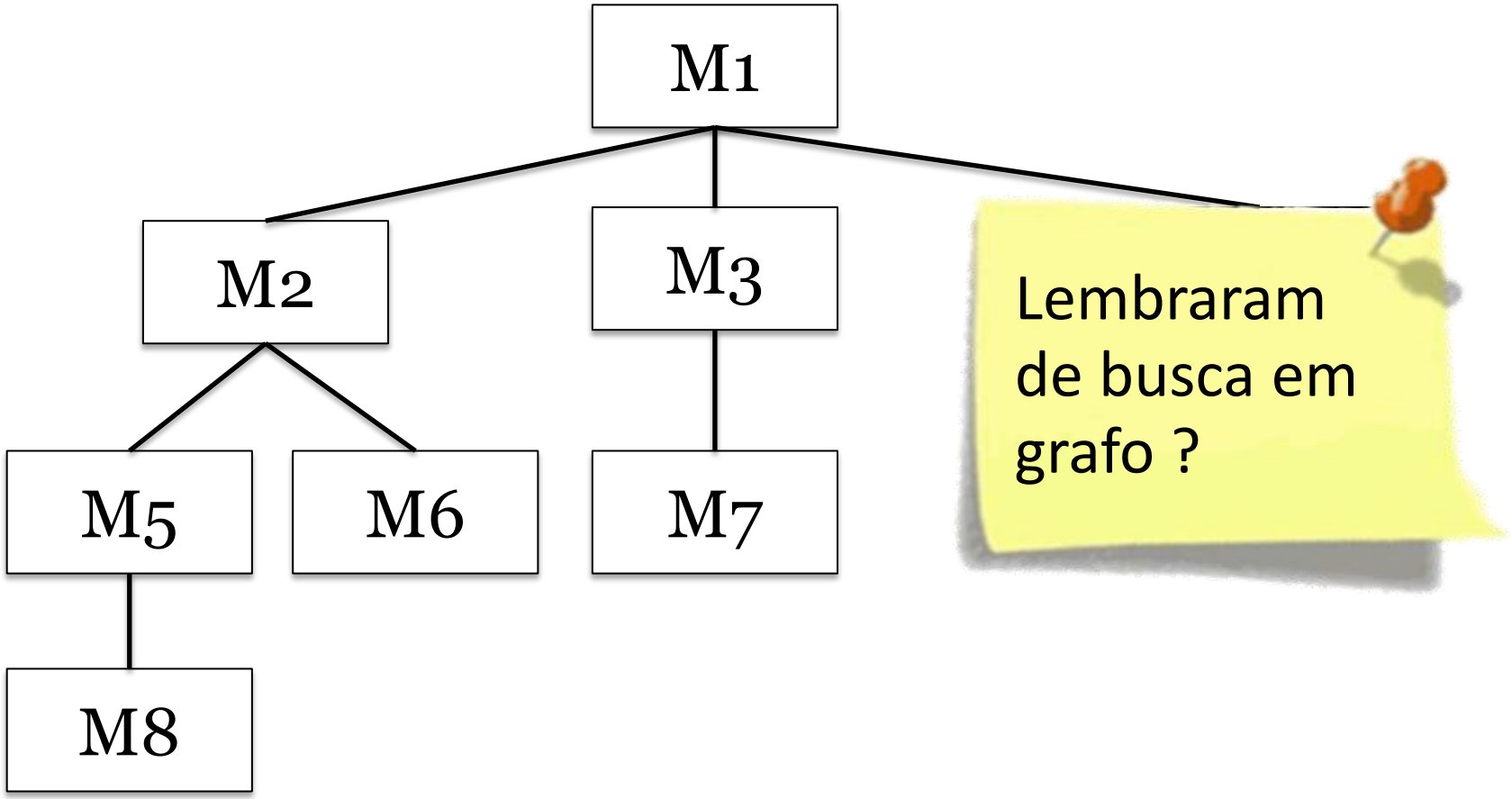
Integração top-down

Primeiro-em-profundidade vs Primeiro-em-largura



Integração top-down

Primeiro-em-profundidade vs Primeiro-em-largura



Integração top-down

1. O modulo principal é usado como pseudocontrolador do teste, e pseudocontrolados sao substituídos pelas unidades subordinadas.
   - Primeiro-em-largura ou primeiro-em-profundidade.
2. Testes sao conduzidos a medida que cada componente é integrado
3. Ao término de cada conjunto de testes, outro pseudocontrolado é substituído pela unidade real.
4. Testes de regressão podem ser conduzidos para garantir que novos erros (efeitos colaterais) não tenham sido introduzidos.

Qual problema com relação a integração top-down ?

Integração top-down

- O processamento de níveis baixos da hierarquia são necessários para testar os níveis superiores. Soluções:
  1. Adiar muitos testes
  2. Desenvolver pseudocontrolados
  3. Integrar o software de baixo para cima.

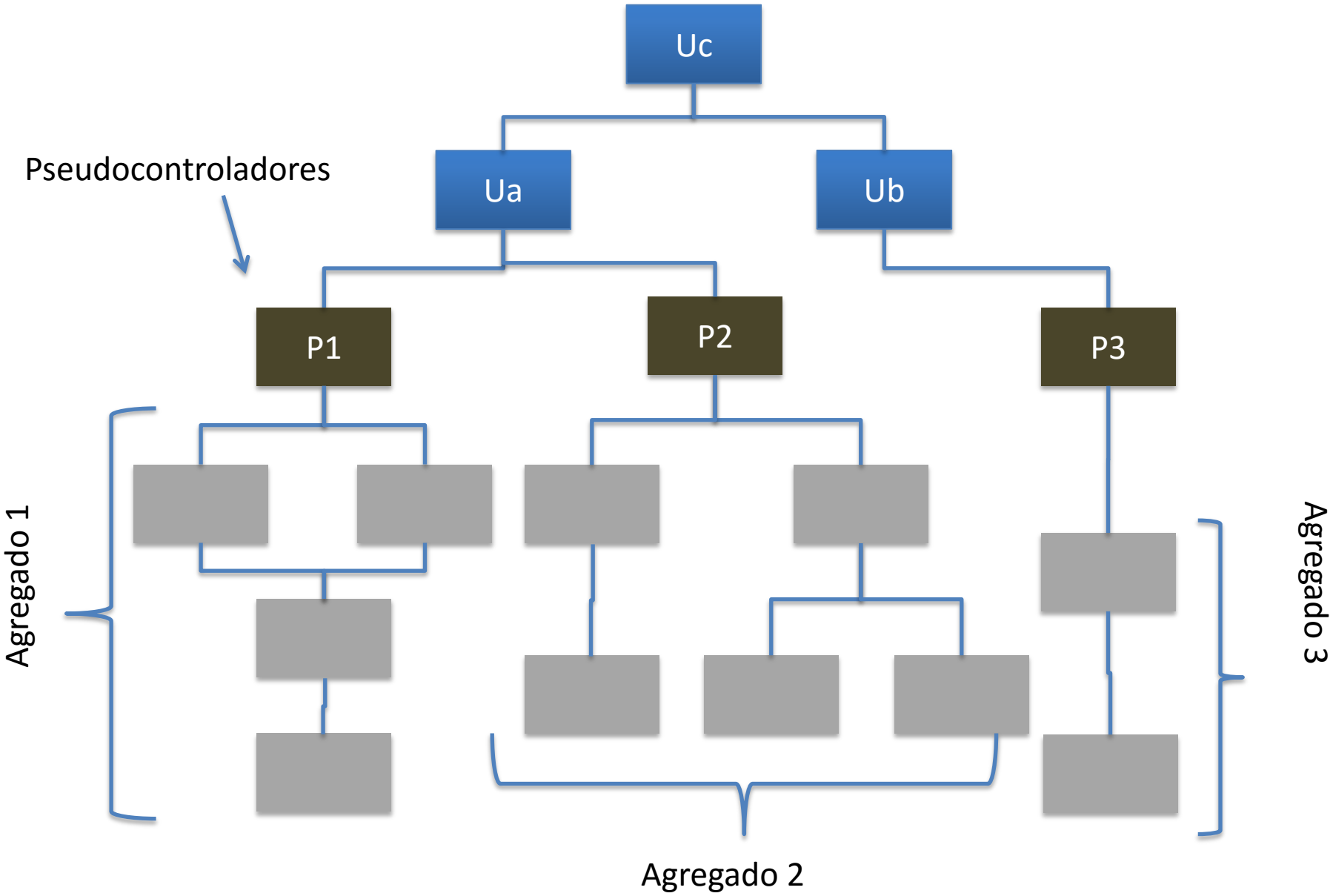
Integração bottom-up

- Os componentes são integrados de baixo para cima, as unidades subordinadas estão sempre disponíveis.
- Necessidades de pseudocontrolados é eliminada.
- Pseudocontroladores são necessários.

Integração bottom-up

1. Componentes de baixo nível são combinados em agregados (clusters, algumas vezes chamados de construções) que realizam uma subfunção especifica.
2. Um pseudocontrolador é escrito para coordenar a entrada e a saída dos casos de teste
3. O agregado e testado.
4. Pseudocontroladores são removidos e agregados são combinados movendo-se para cima da estrutura do programa.

Integraçao bottom-up



E os testes de integração, o uso de orientação a objeto tem algum impacto sobre as abordagens descendentes (top-down) e ascendente (bottom-up) ?

Testes de integração em OO

- Em OO os objetos se relacionam e colaboram em tempo de execução de modo não obvio.
- A abordagem convencional não é efetiva.
- Duas estratégias:
  - Testes baseados na execução, integra um conjunto de classes necessárias para responde uma entrada ou um evento do sistema.
  - Testes baseado no uso uso, testam as classes independentes e depois as dependentes.

Testes de integração em OO

- Em OO os objetos se relacionam e colab ram em tempo de execuç
- A abordagem co
- Duas estratégias
  - Testes baseados nto de classes necessá ou um evento do siste
  - Testes baseado independentes e depois as dependentes.

Observem que o uso de um paradigma tem impacto em todo o processo de desenvolvimento.

Testes de validação

- Começa no fim do teste de integração, quando o software está completamente montado.
- Na validação não existe distinção de paradigmas.
- O teste focaliza ações visíveis ao usuário e saídas do sistema reconhecida pelo usuário.

Testes de validação

- Uma validação se torna bem sucedida, quando o software funciona de modo que pode ser razoavelmente esperado pelo usuário (Pressman).

Testes de validação

- Uma validação se torna bem sucedida, quando o software funciona de modo que pode ser razoavelmente esperado pelo usuário (Pressman).
- Expectativas razoáveis são definidas nas especificações de requisitos de software.

Testes de validação

- Uma validação se torna bem sucedida, quando o softw o que pode ser razoa o usuário (Press
- Expectativas ra as especificações re.

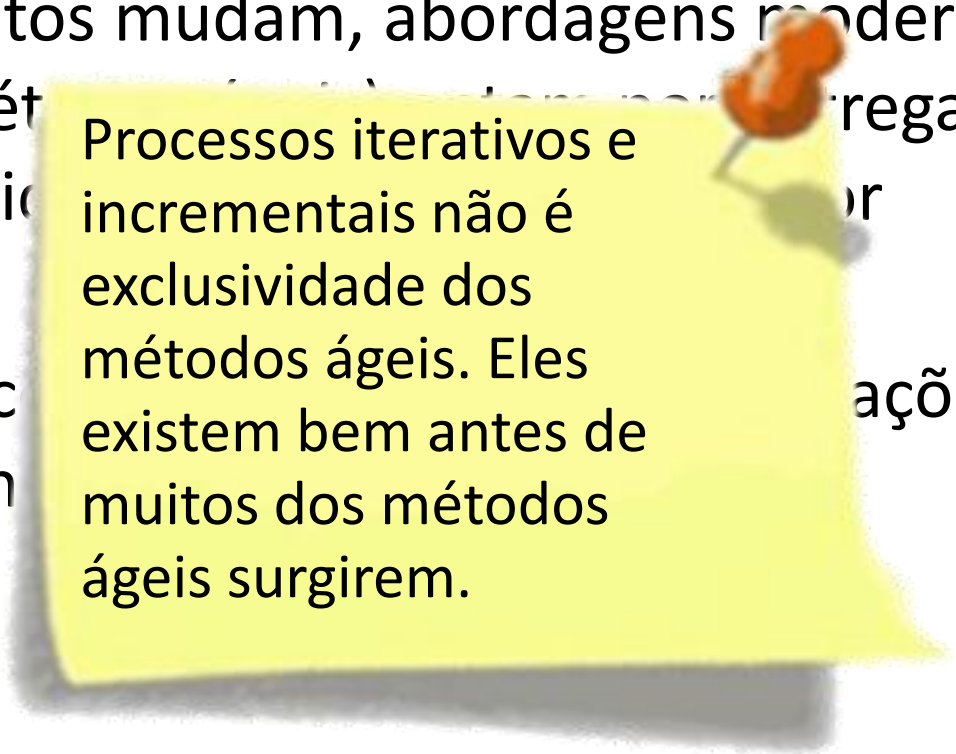
Lembrem-se, requisitos sempre mudam, e dificilmente capturam a real necessidade do usuário.

Testes de validação

- Como os requisitos mudam, abordagens modernas (incluindo os métodos ágeis) optam por entregar incrementos validáveis ao usuário com maior frequência.
- Isso permite encontrar desvios nas especificações mais cedo durante o ciclo do software.

Testes de validação

- Como os requisitos mudam, abordagens modernas (incluindo os mét regar incrementos vali r frequência.
- Isso permite enc ações mais cedo duran



Testes de validação

- Testes com os usuários: teste alfa e beta:

- Teste alfa, o usuário testa o software na presença do desenvolvedor. Os desenvolvedores podem anotar as falhas.

- Testes beta, o usuário testa o software em seu ambiente real. Neste caso os clientes que registram as falhas de software

Testes de sistema

- O software é apenas um elemento de um sistema maior baseado em computador.
  - Hardware, pessoas e informação.
- Testes de sistema é na verdade uma série de diferentes testes, cuja finalidade principal é exercitar por completo o sistema baseado em computador.
  - Testes de recuperação, testes de segurança, testes de estresse e testes de desempenho.

Testes de sistema: Testes de recuperação

- O sistema é tolerante a falha ? Falhas no sistema não podem causar interrupção da função global do sistema.
- Teste de recuperação força um software falhar e verifica:
  - A recuperação é automática?
  - A integridade dos dados são mantidas?
  - O tempo de recuperação (e ou reparo) está em tempo aceitável.

Testes de sistema: teste de segurança

- Testes de segurança verifica se os mecanismos de proteção incorporados a um sistema vão de fato protegê-lo de invasão imprópria.

- O testador desempenha o papel do indíviduo que deseja invadir o sistema. Vale tudo!
  - Tentar obter ou descobrir senhas
  - Usar software especialista para intrusão
  - Causar falhas para tentar invadi-lo durante a recuperação.
  - etc

Testes de sistema: teste de segurança

- Testes de segurança verifica se os mecanismos de proteção incorpo[illegible] sistema v[illegible]de fato protegê-lo de inv[illegible]

- O testador deser[illegible] o que deseja invadir o
  - Tentar obter ou
  - Usar software e
  - Causar falhas para tentar invadi-lo durante a recuperação.
  - etc

O bom teste vai acabar invadindo o sistema. O bom projeto vai tornar o custo maior que o valor da informação a ser obtida.

Teste de estresse

- Testes de estresse executa um sistema de tal forma que a demanda de recursos em quantidade, frequência ou volume são anormais.
- Por exemplo:
  - a) Velocidade de entrada de dados pode ser aumentada, b) casos de teste que exigem o máximo de memória ou outros recursos, c) busca excessivas de dados em disco ...

Teste de estresse

- Testes de estresse executa um sis[illegible]ma de tal forma que a de[illegible]em quantidade, fr[illegible]são anormais.
- Por exemplo:
  - a) Velocidade [illegible] ser aumentada, b) casos de teste que exigem o máximo de memória ou outros recursos, c) busca excessivas em dados em disco ...

O testador irá a todo custo tentar quebrar o sistema.

Teste de desempenho

- O teste de desempenho é projetado para testar o desempenho de um sistema integrado.
- O teste ocorre ao longo de todos os passos do processo de teste.
- Mesmo no nível de unidade pode ser avaliado a medida que testes são conduzidos.
  - O verdadeiro desempenho de um sistema não pode ser avaliado antes que todos os sistemas estejam plenamente integrados.
- Testes de desempenho são frequentemente acoplados a testes de estresse.

Pontos chaves

- Defeitos podem ocasionar a manifestação de erros.
- Erros podem gerar falhas.
- Falha é um comportamento inesperado que afeta diretamente o usuário
- Testes tem como objetivo causar falhas em um programa.
- Depuracao tem como objetivo encontrar um defeito de codificação

Pontos chaves

- Teste é uma atividade relacionada ao processo de verificação e validação, que por sua vez faz parte do processo de garantia de qualidade de software.
- Para ascender ao nivel D do mpsBr, é necessário haver um processo sistemático de testes de software.
- Defeitos de software podem ocorrer em qualquer fase, da especificação a codificação.

Pontos chaves

- Os níveis de testes são: testes de unidade, integração, validação e de sistema.
- Os testes de unidade testam a menor unidade do programa.
- Os testes de integração testam a integração destas unidades.
- Testes de validação testam se o programa se comporta como o esperado pelo usuário.
- Testes de sistema testa o sistema como todo, incluindo hardware, pessoas e informação.

Pontos chaves

- O uso de um paradigma de desenvolvimento afeta o procedimento de testes.
- Testes são muito importantes para garantir a qualidade de um software, porém tem um alto custo.
- Testes realizados de modo informal, tendem ser apenas um custo extra, sem garantia de melhoria na qualidade.
  - Testes devem ser realizados de maneira sistemática