


Laboratório de Engenharia de Software

Princípios de Teste 2

Arndt von Staa
Departamento de Informática
PUC-Rio
Março 2015

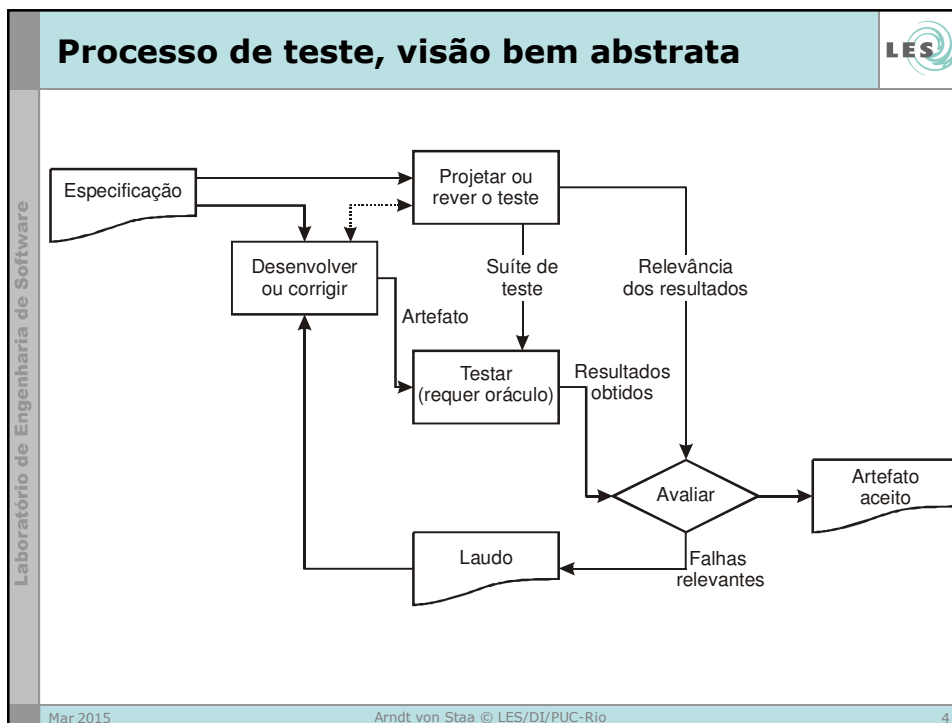
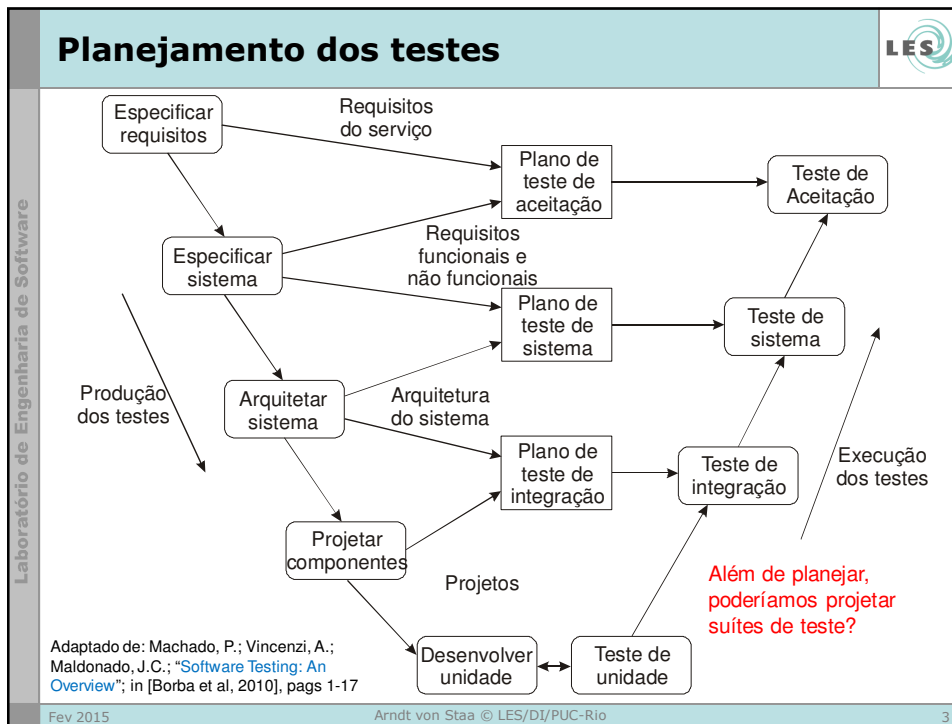
Especificação



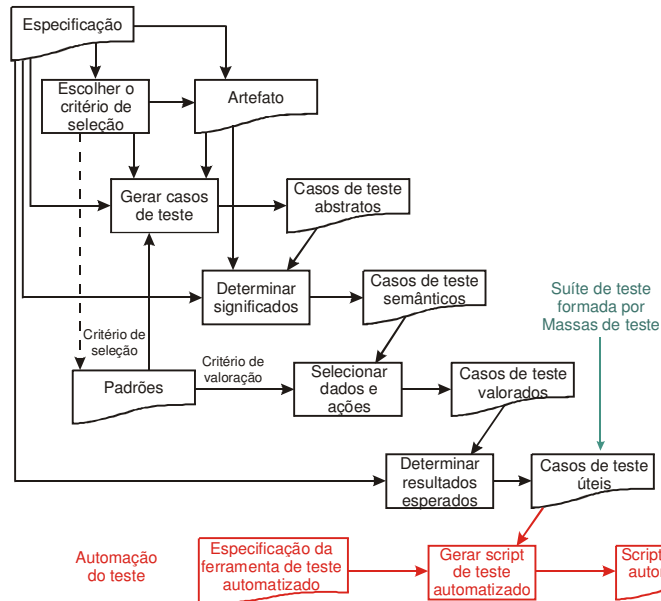
Laboratório de Engenharia de Software

- Objetivo da aula
 - apresentar e discutir os processos relacionados com testes
- Justificativa
 - para realizar testes de forma confiável é necessário trabalhar de forma disciplinada
 - além de testar precisa-se depurar e disponibilizar
 - a disciplina é estabelecida através de processos definidos
- Texto
 - Pezzè, M.; Young, M.; Teste e Análise de Software; Porto Alegre, RS: Bookman; 2008, capítulo 4
 - Bastos, A. 2; Rios, E.; Cristalli, R.; Moreira, T.; *Base de Conhecimento em Teste de Software*; São Paulo: Martins; 2007

Mar 2015Arndt von Staa © LES/DI/PUC-Rio2



Processo de seleção de casos de teste



Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

5

Processo de seleção de casos de teste



```

while ( ! feof( pArq ))
{
    ...
    printf( "%d" , strlen( Buffer ) ) ;
    ...
}
    
```

- Casos de teste:

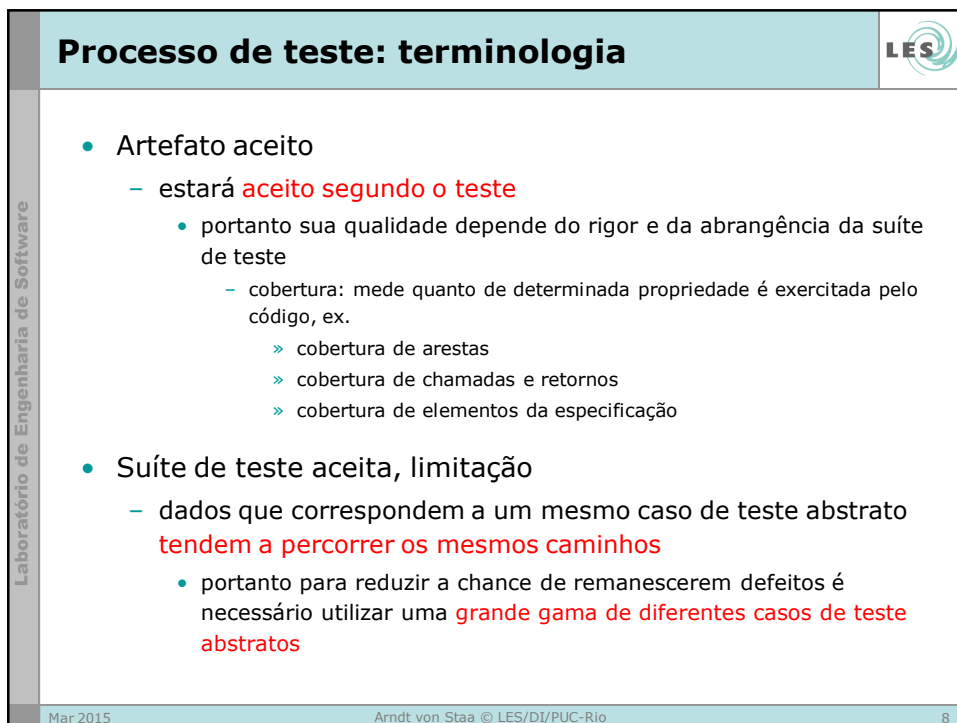
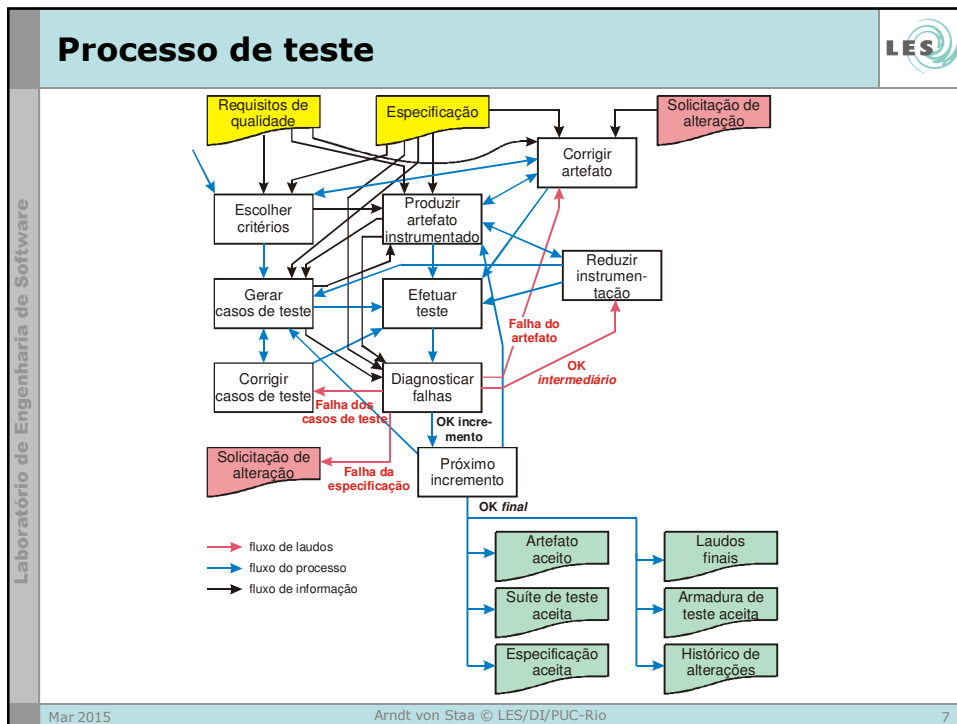
casos de teste úteis

Abstratos	Semânticos	Valorados	Resultado esperado
Executa 0 vezes	Arquivo vazio	{ }	<i>nada</i>
Executa 1 vez	Arquivo com 1 registro	{ aaa }	3
Executa 3 vezes	Arquivo com 3 registros	{ aaa , bb , c }	3 2 1

Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

6



Processo de teste: terminologia



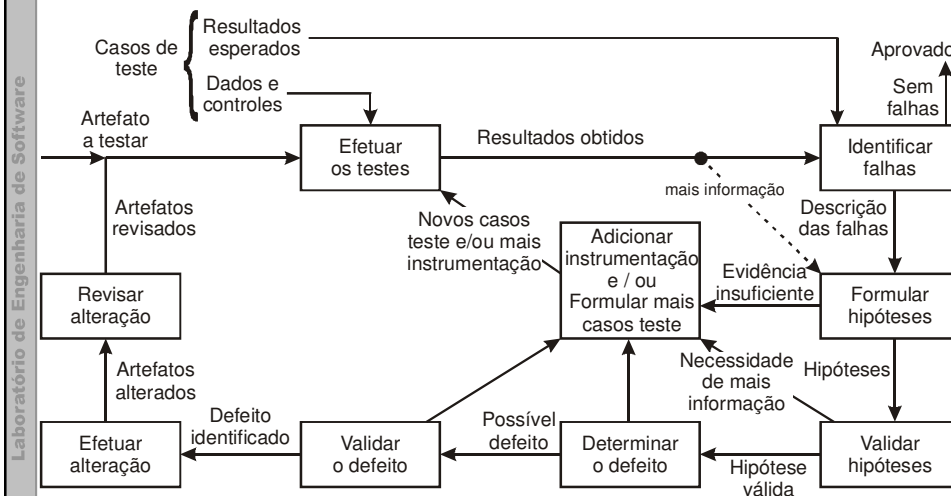
- Especificação aceita
 - estará aceita se estiver **em conformidade com os testes**
 - portanto existe interdependência bidirecional – a especificação implica o teste e este implica aquela
 - possivelmente existem **solicitações de alteração pendentes**
 - portanto o artefato estará aceito com vistas a uma **versão da especificação** possivelmente diferente da correntemente disponível
- História da evolução
 - registros no sistema de controle de versão
 - permitem identificar as causas recorrentes de problemas e quais os módulos problemáticos, desde que se siga um padrão de comportamento adequado
 - cada pendência deve ser tratada de forma independente das demais
 - ao terminar uma pendência todas as correspondentes alterações são registradas no controle de versão

Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

9

Processo de depuração



Mar 2015

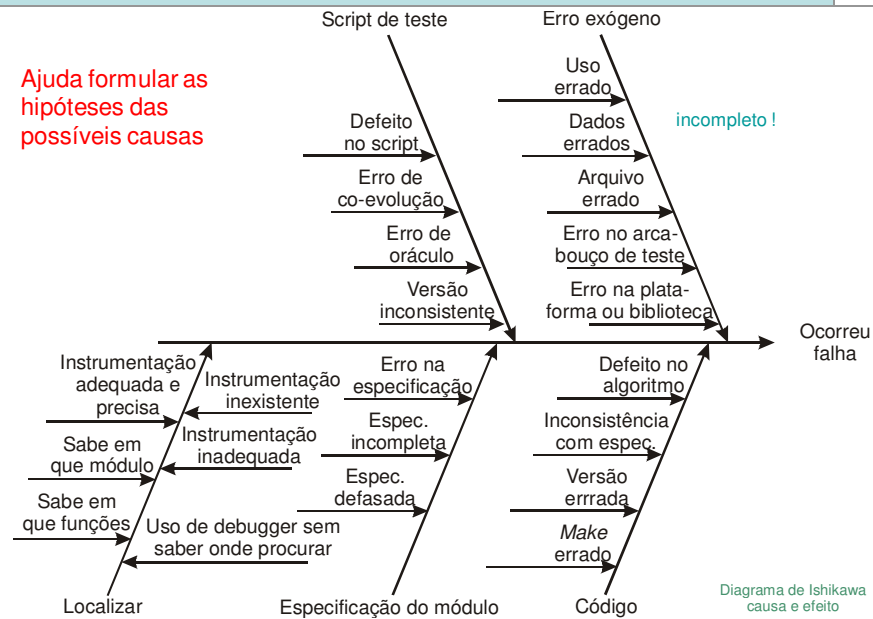
Arndt von Staa © LES/DI/PUC-Rio

10

Diagramas de causa e efeito - Ishikawa



Ajuda formular as hipóteses das possíveis causas



Mar 2015

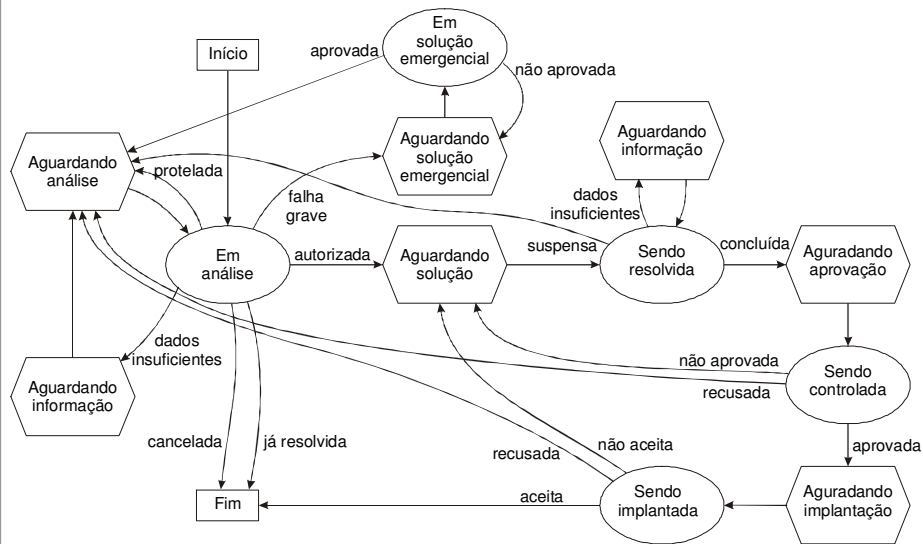
Arndt von Staa © LES/DI/PUC-Rio

11

Acompanhamento de problemas (recordação)



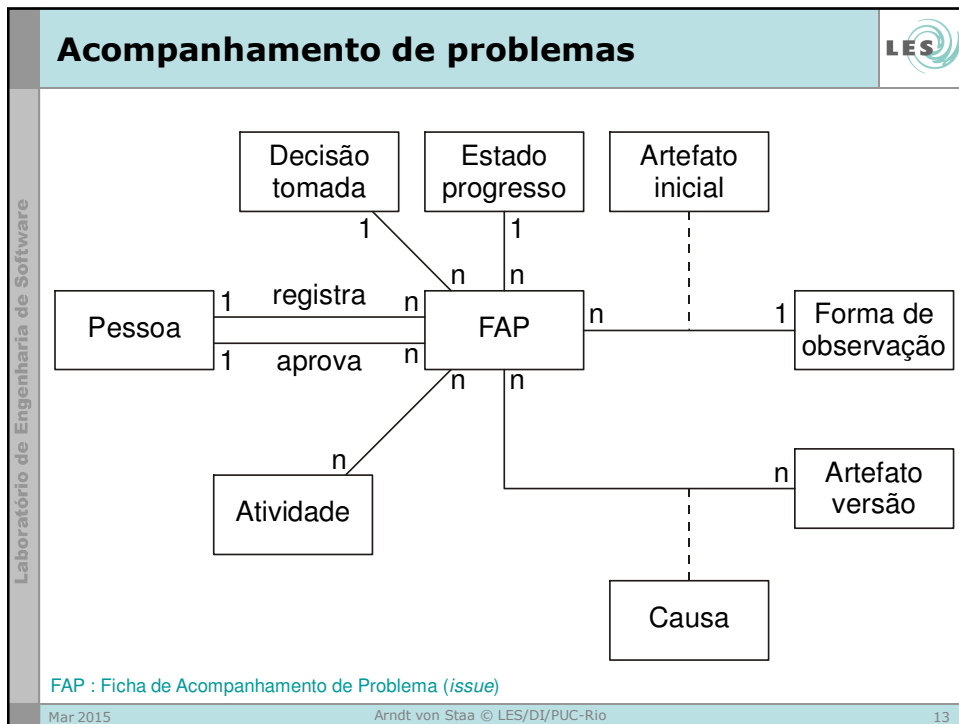
esboço




Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

12



Teste é somente parte do problema




Laboratório de Engenharia de Software

- O objetivo de registrar falhas é **promover a remoção dos defeitos causadores** de forma completa e correta
 - a **remoção completa e correta** do defeito que gerou a falha é que conduz à **melhoria da qualidade** do artefato

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
14

Laboratório de Engenharia de Software

Teste é somente parte do problema




- Tarefas a serem realizadas em conjunção com o teste
 - **detectar**: identificar que ocorreu uma falha (**observar um erro**)
 - realizado pelos oráculos do teste
 - oráculos podem ser instrumentos adicionados ao código
 - **diagnosticar**: identificar *corretamente* o defeito a partir da falha
 - procurar a **causa** da falha
 - pode ser caso de teste errado, cenário errado, ...
 - **depurar**: eliminar *completamente* o defeito, sem adicionar novos
 - **retestar**: verificar se a nova versão do artefato satisfaz a sua nova especificação e/ou sua correção
 - **teste de regressão**: verificar se tudo que não foi alterado continua operando corretamente
 - **(re)disponibilizar**: tornar a nova versão disponível para os interessados
 - **(re)instalar** (implantar): por a nova versão em operação

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
15

Laboratório de Engenharia de Software

Relato de falhas



- O melhor testador
 - não é aquele que encontra o maior número de falhas
 - é aquele que consegue **promover a eliminação do maior número dos defeitos** correspondentes às falhas e vulnerabilidades por ele encontrado
- O relato de falha (laudo, FAP) é o instrumento que se utiliza para **vender a falha** ao programador responsável por eliminar o defeito causador
 - um exemplo de modelo para relatar falhas:
<https://bugs.opera.com/wizard/>


Kaner, C.; *Bug Advocacy: How to Win Friends and Stomp BUGs*; 2000; Buscado em: 2008; URL: <http://www.kaner.com/pdfs/bugadvoc.pdf>

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
16

Laboratório de Engenharia de Software

Relato de falhas

FAP – Ficha de acompanhamento de problemas




- Conteúdo do **relato de falha (laudo)** [Kaner, 1993]:
 - data
 - quem relatou
 - em que programa ocorreu?
 - versão
 - se tiver, o **número do build**
 - resumo da falha
 - um parágrafo (título) de no máximo 2 linhas
 - tipo da falha (ver a seguir)
 - severidade da falha (ver a seguir)
 - descrição
 - o que ocorreu e o que deveria ter acontecido?
 - o que você estava fazendo quando ocorreu a falha?
 - você pode fornecer os dados que estava usando?
 - cenário de uso ao observar a falha
 - como replicar a falha?
 - opcionalmente, sugestão de solução
 - anexos

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
17

Laboratório de Engenharia de Software

Relato de falhas


FAP – Ficha de acompanhamento de problemas




Tipo da falha

- erro de processamento
- erro de projeto / especificação
- erro de documentação
- erro de biblioteca
- erro de plataforma de software
- erro de hardware
- dúvida
- . . .


Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
18

Laboratório de Engenharia de Software	Relato de falhas FAP – Ficha de acompanhamento de problemas		
	<p>Severidade</p> <ul style="list-style-type: none"> • problema menor <ul style="list-style-type: none"> – dá para ser contornado – é uma inconveniência • grave <ul style="list-style-type: none"> – não dá para continuar usando o artefato • infeccioso <ul style="list-style-type: none"> – propaga erros para outros programas ou sistemas • catástrofe <ul style="list-style-type: none"> – destrói dados persistentes – provoca grandes danos – provoca quebra de equipamento – provoca mortes, ruína de empresas, desastres ecológicos 		
Mar 2015		Arndt von Staa © LES/DI/PUC-Rio	19

Laboratório de Engenharia de Software	Relato de falhas – solução dada FAP – Ficha de acompanhamento de problemas		
	<ul style="list-style-type: none"> • Conteúdo do relato da solução dada <ul style="list-style-type: none"> – Estados e datas: aberto, em resolução, em implantação, resolvido – Responsável por realizar (gerenciar) a alteração – Natureza da decisão <ul style="list-style-type: none"> • pendente de decisão, não reprodutível, resolvido, concluído, não é falha, cancelado pelo informante, requer mais informação, duplicata, cancelada, postergada, ... – Descrição da solução – Artefatos alterados e versão da alteração <ul style="list-style-type: none"> • natureza da alteração, por artefato <ul style="list-style-type: none"> – código simples, código complexo, falta de controle, reestruturação, projeto, arquitetura, especificação – Responsável por aceitar a solução 		
Mar 2015		Arndt von Staa © LES/DI/PUC-Rio	20

Laboratório de Engenharia de Software

Manutenção




- Causas para a manutenção
 - Melhoria (evolutiva)
 - modifica a funcionalidade do artefato
 - velho 50% Linux 12% novo 65%
 - Adaptativa [Schach et al, 2003] [Nosek & Palvia, 1990]
 - modifica o artefato sem afetar a sua funcionalidade
 - velho 20% Linux 0% novo 18%
 - Corretiva
 - remove defeitos
 - velho 25% Linux 87% novo 17%
 - Perfectiva / outros
 - remove deficiências
 - velho 5% Linux 1% novo --
- Schach, S.; Jin, B.; Yu, L.; Heller, G. Z.; "Determining the Distribution of Maintenance Categories: Survey versus Measurement"; *Empirical Software Engineering* 8; Kluwer; 2003; pp 351–365
- Nosek, J. T.; Palvia, P.; "Software Maintenance Management: changes in the last decade"; *Software Maintenance: Research and Practice* 2(3); 1990; pp 157-174

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
21

Laboratório de Engenharia de Software

Manutenção



- Ações de manutenção
 - adiciona artefato
 - exclui artefato
 - adiciona, altera, exclui elementos de artefato
 - declaração e manipulação (acesso) de dados
 - lógica e estrutura (*refactoring?*)
 - processamento (cálculo, fórmulas)
 - inicialização
 - interface humano-computador
 - interface de módulo
 - documentação papel
 - documentação help
 - documentação tutoriais
 - outros

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
22

LES

- Laboratório de Engenharia de Software

23


LES



24

Laboratório de Engenharia de Software

Fatores de qualidade visando manutenção




- **testabilidade**: facilidade de ser retestado com suficiente rigor e abrangência sempre que necessário
 - torna necessário que o artefato seja acompanhado de
 - massas de teste aceitas
 - arcabouço de teste aceito
 - módulos de teste aceitos
 - ferramentas de desenvolvimento utilizadas
 - artefatos de desenvolvimento
 - produtos são artefatos de desenvolvimento entregues ao usuário ou cliente

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
25

Laboratório de Engenharia de Software

Fatores de qualidade visando manutenção




- **detectabilidade**: facilidade de identificar que ocorreu um erro (falha)
 - erros ocorrem
 - **endógeno** – por causas internas: defeitos, vulnerabilidades
 - **exógeno** – por causas externas
 - requer **redundâncias** contidas no sistema (código), exemplos
 - instrumentação que efetivamente ajude
 - inserção de **assertivas** inseridas na forma de **instrumentação**
 - realização de “**prova real**” – efetuar uma operação inversa e verificar se reconstruiu os dados de entrada
 - **comparar n resultados** (réplicas) e determinar o potencial responsável pelas discrepâncias observadas:
 - uso de diferentes especificações para calcular o resultado
 - implementar n versões de uma mesma especificação
 - distribuir v >= 1 versões sobre n >= 3 máquinas independentes
 - . . .

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
26

Laboratório de Engenharia de Software

Fatores de qualidade visando manutenção

- **diagnosticabilidade**: facilidade de identificar o defeito a partir da falha
 - requer que as falhas sejam acompanhadas de **informação complementar** descrevendo o estado da execução no **ponto em que foi observado o erro** (**sinalizada a falha**), exemplos:
 - pilha de execução
 - estado das variáveis relevantes
 - ações e dados fornecidos pelo usuário
 - estado dos recursos funcionais, ex. sockets, semáforos, ...
 - estado dos sensores e atuadores
 - idealmente o ponto de observação **deve estar muito próximo do defeito causador**
 - isso implica que a detecção deve ser realizada por instrumentação
 - o ser humano não tem condições para sinalizar falhas com a granularidade necessária




Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
27

Laboratório de Engenharia de Software


Fatores de qualidade visando manutenção

- **depurabilidade**: facilidade de eliminar completa e corretamente o defeito
 - artefato com boa qualidade de engenharia
 - especificação, arquitetura e projeto – correspondem exatamente a o que está implementado
 - documentação técnica suficiente para o mantenedor localizar e entender como alterar o defeito
 - ambiente utilizado para desenvolver existe
 - sub-sistema de apoio à manutenção existe
 - scripts de teste automatizado
 - roteiros de teste
 - scripts de recompilação e integração
 - documentação técnica útil



Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
28

Ferramentas recomendadas




Laboratório de Engenharia de Software

Nível mínimo:

- **Ambiente de desenvolvimento integrado** (IDE – *Integrated Development Environment*)
 - Eclipse, Visual Studio, ...
- **Sistema de controle de versões**
 - SVS, Subversion, GIT, ...
- **Sistema automatizado para a reconstrução de programas**
 - make, gmake, ant, maven, hudson ...
- **Sistema de acompanhamento de problemas**
 - Bugzilla, Jira, ...
- **Repositório compartilhado de artefatos aceitos**
- **Repositório de padrões, diretrizes, e convenções**
- **Repositório de documentação técnica** (JavaDoc ou similar)

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
29

Ferramentas recomendadas



Laboratório de Engenharia de Software


Nível intermediário

- Todas as do nível mínimo
- **Sistema de gerência de requisitos**
 - acompanha a alteração de requisitos
 - permite determinar o impacto de alterações
 - rastreia os requisitos nos artefatos desenvolvidos
- **Sistema de teste automatizado básico**
 - realiza o teste de unidades
 - funções, classes, módulos
 - realiza o teste de regressão das unidades

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
30

Laboratório de Engenharia de Software

Ferramentas recomendadas




Nível avançado

- Todos do nível intermediário
- **Transformadores**
 - geram esqueletos a partir de diagramas
 - geram esqueletos de módulos de teste
 - geram outros artefatos intermediários
- **Geradores**
 - geram artefatos prontos para serem utilizados (compilados)
 - geram módulos de teste
 - geram casos de teste úteis *idealmente scripts de teste*
- **Analísadores estáticos**
 - verificam propriedades de artefatos segundo regras definidas
- **Teste automatizado avançado**
 - junit, dbUnit, jBehave, selenium, concordion, ...

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
31

Laboratório de Engenharia de Software

Bibliografia complementar



- Huizinga, D.; Kolawa, A.; *Automated Defect Prevention: Best Practices in Software Management*; Hoboken, New Jersey: John Wiley & Sons; 2007
- Kaner, C.; Falk, J.; Nguyen, H.Q.; *Testing Computer Software*; International Thompson Computer Press; 1993
- Kaner, C.; *Bug Advocacy: How to Win Friends and Stomp BUGs*; 2000; www.kaner.com (testing website) www.badsoftware.com (legal website)
- Kemerer, C.F.; Slaughter, S.; "An Empirical Approach to Studying Software Evolution"; *IEEE Transactions on Software Engineering* 28(4); 1999; pages 493-509
- Lewis, W.E.; *Software Testing and Continuous Quality Improvement*; Boca Raton: Auerbach; 2000

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
32

Laboratório de Engenharia de Software

LES

FIM

Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

33