


Laboratório de Engenharia de Software

# Técnicas de controle da qualidade

Arndt von Staa  
Departamento de Informática  
PUC-Rio  
Fevereiro 2015

## Especificação



Laboratório de Engenharia de Software

- Objetivo da aula
  - Discutir, de um ponto de vista macroscópico, as atividades e técnicas relacionadas com o controle da qualidade
- Justificativa
  - uma visão abrangente das técnicas de controle da qualidade facilita a compreensão da interdependência entre elas e com os processos de desenvolvimento

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

## Processo de desenvolvimento



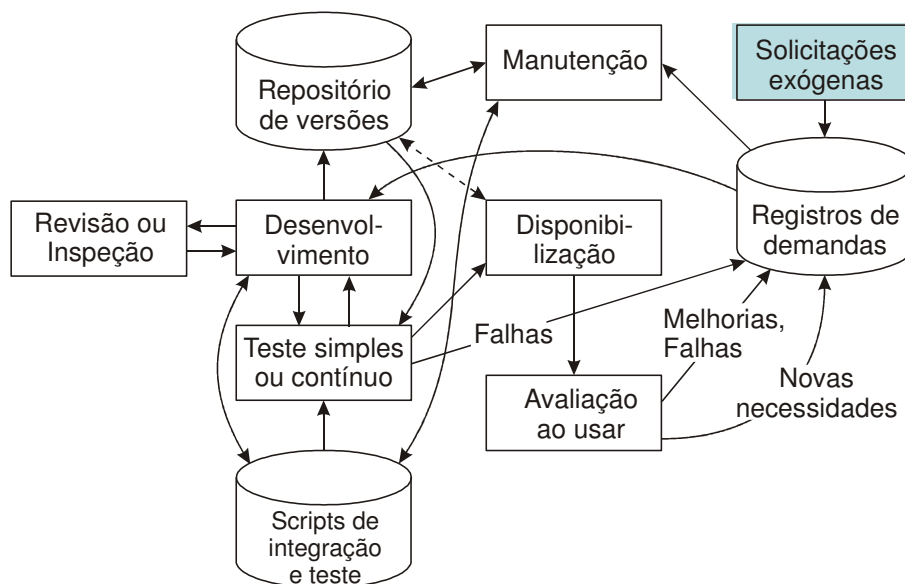
- Software, por mais que nos esforcemos, conterá defeitos, portanto poderá falhar.
  - o que você sugere fazer para poder assegurar que o software terá pelo menos qualidade por desenvolvimento satisfatória?
  - o que você sugere fazer para poder assegurar que o software terá pelo menos qualidade por manutenção satisfatória?

Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

3

## Ciclo de vida total (bem simplificado)



Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

4

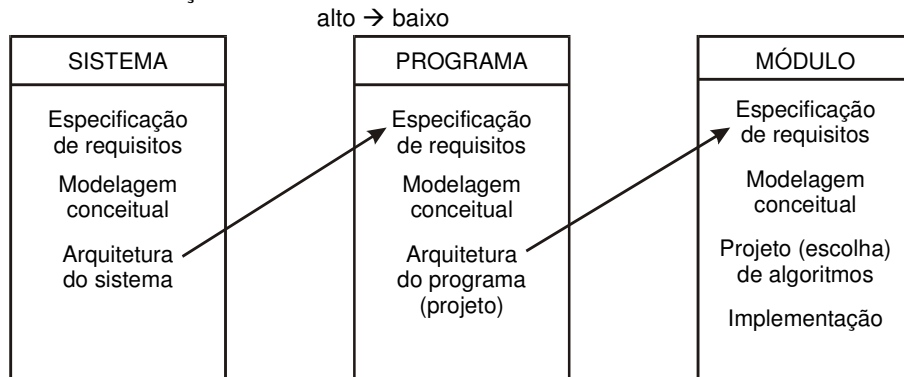
## Evolução da especificação



Especificação ocorre quase continuamente, varia:

- o nível de abstração
- a tecnologia utilizada ao especificar

Nível de abstração:

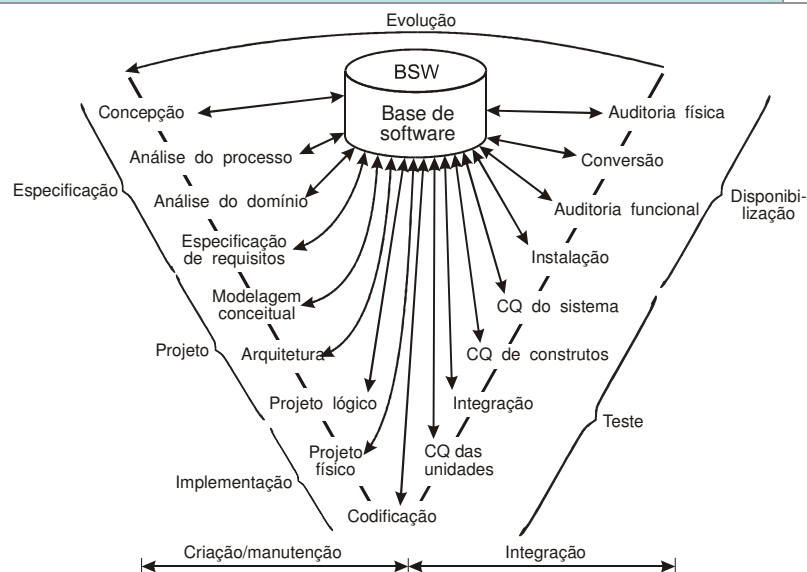


Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

5

## Etapas do processo de desenvolvimento

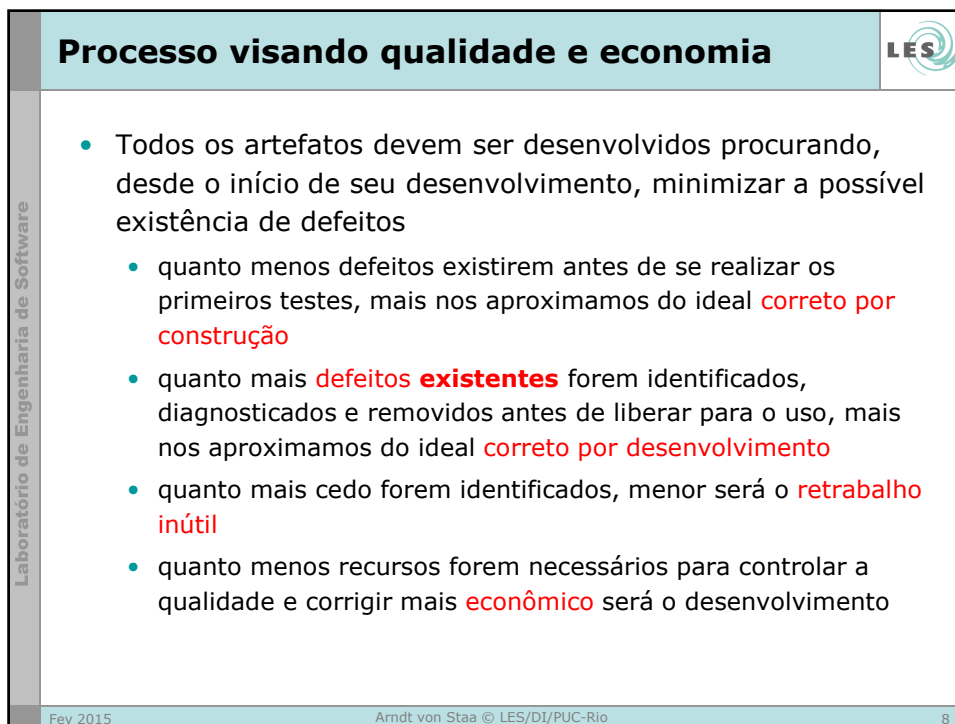
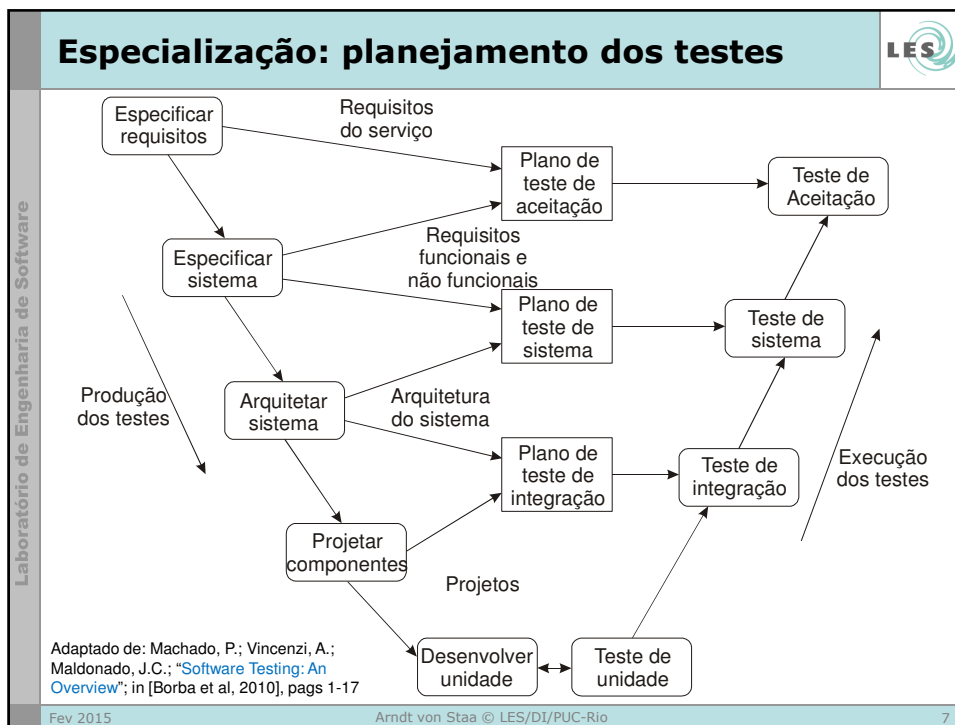


Staa, A.v.; *Programação Modular*; Rio de Janeiro: Campus; 2000; capítulos 2 e 10

Fev 2015


Arndt von Staa © LES/DI/PUC-Rio

6



Laboratório de Engenharia de Software

## Objetivos do controle




- O objetivo do controle da qualidade é **identificar discrepâncias** com relação aos *interesses explícitos e implícitos dos usuários e clientes*, com relação a
  - requisitos **funcionais, não funcionais**, de **interface humano-sistema** e outros
  - o controle deve ser realizado considerando todos os artefatos
    - requisitos
    - arquitetura
    - projeto da interface humano-sistema (*look and feel*)
    - projeto lógico
    - suítes de teste
    - código
    - documentação para o usuário
    - . . .

IHC <= IHS – interface humano-sistema é mais amplo, pois engloba também as interações que não são estritamente dirigidas a computadores

Fev 2015
Arndt von Staa © LES/DI/PUC-Rio
9

Laboratório de Engenharia de Software

## Objetivos do controle: usuários e clientes




- Identificar (é possível dizer “todas as” ?) discrepâncias com relação ao que é **esperado pelos usuários e clientes**
  - **serviços de fato prestados** pelo artefato
  - confronto dos serviços com o desejado pelo usuário
    - verificar se é a implementação fidedigna (correta e completa) do **problema correto**
  - interface com o usuário
    - facilidade de uso
    - facilidade de aprendizado
    - adequação à cultura do usuário
    - segurança
    - capacidade de tolerar e recuperar de erros do usuário (clemência)
  - baixo custo considerando a vida útil
    - CTP – *custo total de propriedade* ( TCO – *total cost of ownership* )
  - . . .

Fev 2015
Arndt von Staa © LES/DI/PUC-Rio
10

Laboratório de Engenharia de Software

## Objetivos do controle: especificação

- Identificar discrepâncias com relação à especificação do artefato
  - requisitos funcionais
    - relacionados ao domínio da aplicação
  - requisitos de interface com o usuário
  - requisitos não funcionais
    - em geral são os requisitos de qualidade
      - segurança
      - capacidade de processamento
      - escalabilidade
      - ...
  - requisitos inversos
    - coisas que não podem acontecer
  - condições de aceitação
  - condições contratuais




Fev 2015
Arndt von Staa © LES/DI/PUC-Rio
11

Laboratório de Engenharia de Software


## Propriedades do controle: perfeição?

- O controle da qualidade é
  - um filtro imperfeito
    - identifica somente uma parte dos problemas existentes
      - defeitos, deficiências e vulnerabilidades observadas
    - os demais problemas permanecem e são desconhecidos
      - defeitos, deficiências e vulnerabilidades remanescentes
    - na indústria hoje software muito bom tem cerca de 1 defeito para cada 1.000 linhas de código
      - mas pouco software é muito bom ...
      - o estado da prática precisa melhorar muito!
      - existe gente que trabalha com algo próximo a 1 / 10.000 LOC
  - problema da existência: podemos procurar e encontrar defeitos, mas se não encontrarmos não podemos concluir que não existam

- estado da prática      – o que se pratica costumeiramente nas organizações
- estado da arte        – o nível de desenvolvimento mais avançado que se consegue com o conhecimento atual




Fev 2015
Arndt von Staa © LES/DI/PUC-Rio
12

**Contexto**


Laboratório de Engenharia de Software

- O grau de qualidade a ser atingido depende do contexto:
  - natureza do sistema, ex.
    - sistema de busca de informação
    - sistema de controle de processo contínuo
  - existência de legislação a ser respeitada, ex.
    - certificação de software automotivo
  - consequências de eventuais erros, ex.
    - aborrecimentos
    - perda de vida
  - tempo de vida esperado, ex.
    - a ser usado poucas vezes, ou por pouco tempo
    - a ser usado por longo tempo
    - a ser usado e substituído por similar durante a existência da organização cliente

Fev 2015
Arndt von Staa © LES/DI/PUC-Rio
13

**Propriedades do controle**


Laboratório de Engenharia de Software


- Cada técnica de controle da qualidade possui seu **domínio de eficácia**
  - classe de defeitos mais naturalmente encontrados ao aplicar a técnica
- Como reduzir a chance de defeitos existentes passarem despercebidos?
  - utilize diversas técnicas de controle da qualidade
  - procure usar técnicas rigorosas, ex. argumentação da corretude
    - cuidado com excesso de zelo
    - muitas técnicas rigorosas são caras e não trazem benefícios compatíveis
  - utilize instrumentação para controlar a integridade em tempo de execução

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

Laboratório de Engenharia de Software

## Propriedades do controle: resultado

- O resultado do controle da qualidade é um **laudo**
  - **relaciona os problemas identificados, falhas, incidentes**
- O laudo pode assumir diversas formas
  - **relatório** em **formato livre** relacionado com o artefato
  - **anotações** no próprio artefato
    - ex. acompanhamento de alterações do Word
  - **log gerado** por ferramentas de controle da qualidade
  - **caderno** de registro de problemas
  - listas de **pendências** (*to do lists, backlog*)
    - ferramentas de registro e acompanhamento: Bugzilla, Jira, ...
  - conjunto de **fichas de acompanhamento de problema**
  - rascunhos, lembranças → isso é **péssimo** ☹




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

Laboratório de Engenharia de Software

## Propriedades do controle: assegura o que?

- Controle da qualidade **não assegura** qualidade!
  - controle da qualidade **não corrige!**
    - como já foi dito, o resultado do controle é um **laudo**
    - controle da qualidade somente verifica o **quanto o artefato se aproxima da qualidade** desejada
    - procura **encontrar e relatar** defeitos, deficiências ou falhas
- Entretanto, saber como será controlado **antes de desenvolver** induz o desenvolvedor a se **aproximar, por construção, da qualidade requerida** pelo controle



- Weinberg, G.M.; *The Psychology of Computer Programming*; 2nd edition; Dorset House; 1998  
Obs. a primeira versão foi publicada há mais de 30 anos...

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



## Propriedades do controle



- Controle da qualidade **não assegura** qualidade!
- Porém, a **solução dos problemas** identificados no laudo **pode** levar à **melhoria** da qualidade
  - Cabe à equipe de desenvolvimento resolver os problemas indicados pela gerência
    - de forma **completa**
      - **diagnose** para determinar a **causa exata** (defeito) da falha
      - **depuração** para eliminar completamente a causa
    - **sem gerar novos** defeitos !!!
    - procurando aprender com os erros, para evitar cometê-los de novo

Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

17

## Propriedades do controle: planejamento?



- O controle da qualidade de cada artefato deve ser planejado (definido) **junto com a especificação, arquitetura, projeto e codificação**
  - que padrões e normas devem ser obedecidos?
  - como serão verificadas as especificações?
    - as especificações são verificáveis? São testáveis?
  - que controles e quando devem ser aplicados?
  - que ferramentas serão utilizadas?
  - que instrumentação e quando deve ser incluída?
  - como será testado?
    - plano de teste
  - como será aceito?
    - quais são os critérios de aceitação?
    - quando sei que testei o suficiente?


Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

18

Laboratório de Engenharia de Software

## Propriedades do controle




- *Corolário*: todos os itens das especificações e dos padrões precisam **ser verificáveis!**
  - é verificável se for **possível mostrar o que vale e o que NÃO vale**
    - inclusive os requisitos que tratam de qualidade
  - sem dispor de uma especificação verificável como posso dizer **racionalmente** o que seria aceitável?
  - implica a **manutenção contínua** (co-evolução) das especificações
- *Ideal*: todos os itens das especificações funcionais e não funcionais deveriam ser **testáveis**
  - de preferência de forma automática
  - teste automático é uma forma verificável de **especificação através de exemplos**

Fev 2015
Arndt von Staa © LES/DI/PUC-Rio
19

Laboratório de Engenharia de Software

## Propriedades do controle




- Após cada alteração é necessário repetir o controle da qualidade
  - isso compromete a produtividade
  - gera perdas devido ao retrabalho
    - retrabalho inútil, quando se trata de correção

Custos? Como evitar que cresçam demasiadamente?

Fev 2015
Arndt von Staa © LES/DI/PUC-Rio
20

Laboratório de Engenharia de Software

## Propriedades do controle



- Sugestão: **automatizar** o controle da qualidade
  - como fazê-lo com relação a código?
    - é possível ser feito para tudo o que é código?
    - i.e. é possível automatizar o teste de todos os aspectos do código?
  - como fazê-lo com coisas que não são código?
    - ex. consistência entre tutoriais e *help* e a implementação
  - como fazê-lo com relação a
 

- arquitetura
    - projeto
    - modelos
    - ...

?


Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

21

Laboratório de Engenharia de Software

## Controle passo a passo

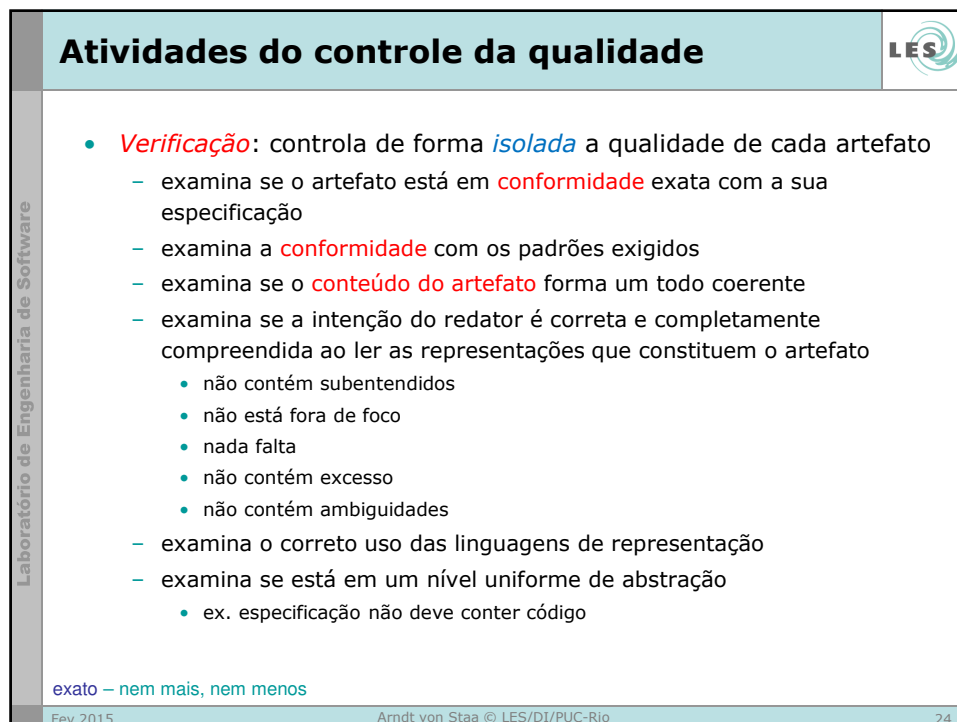
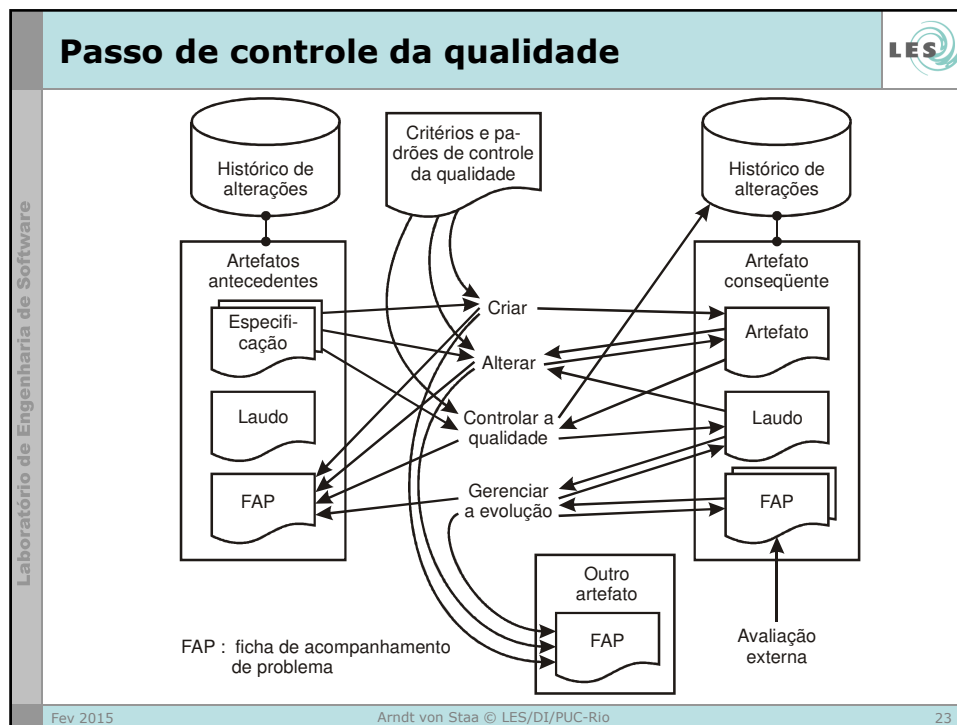


- Deve-se assegurar **continuamente a coerência** entre todos os artefatos
- O controle da qualidade deve envolver **todos os tipos de artefatos**, ex.
  - programas
  - documentos para o usuário
    - inclusive os de marketing
  - sistemas de instalação
  - sistema de auxílio
  - tutoriais
  - bases de dados inicializadas
  - arquivos de dados persistentes
  - ...

Fev 2015


Arndt von Staa © LES/DI/PUC-Rio

22



Laboratório de Engenharia de Software

## Atividades do controle da qualidade




- **Validação**: controla a qualidade dos *inter-relacionamentos* entre artefatos
  - examina se não existem conflitos com outros artefatos
    - em especial com relação às especificações
  - examina se o **conjunto de artefatos forma um todo coerente**
  - examina se todas as interfaces entre artefatos são respeitadas

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

Laboratório de Engenharia de Software

## Atividades do controle da qualidade



- Se passou pela verificação e pela validação, o artefato estará **correto com relação** à sua especificação e a outros artefatos, **segundo a forma de controle usada**.
  - implementação correta do **problema especificado**
  - infelizmente se a especificação não estiver correta: pode levar à **implementação correta do problema errado**
- **para o usuário estará errado, tanto faz a causa**

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

## Atividades do controle da qualidade



- **Aprovação**: controla a qualidade do artefato com relação às **atuais** necessidades e expectativas explícitas e implícitas dos usuários
  - para especificações (tudo o que está no lado criação do "V"):
    - examina se a solução proposta **poderá vir a** atender às **atuais** necessidades e expectativas do usuário
  - para implementações (tudo que está no lado integração do "V"):
    - examina se o artefato **efetivamente atende** às **atuais** necessidades, expectativas do usuário ou potenciais usuários
  - para **antecipar a aprovação** é recomendado desenvolver de forma **incremental**
    - ao término de cada incremento examina-se a satisfação das necessidades explícitas e implícitas dos usuários
    - implica que cada incremento leve a alguma coisa útil

Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

27

## Atividades do controle da qualidade




- Se passou pelas três: *verificação, validação e aprovação*, o artefato será, **em princípio**, uma implementação correta do **problema correto**
  - por que o pé atrás: "em princípio"?

Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

28

## Laudo, registro de problema




Laboratório de Engenharia de Software

- Artefato (construto)
  - nome
  - versão
  - data
  - quem
  - como:
    - revisão, inspeção
    - caso de teste
    - uso
    - outros, ex. desenvolvimento de outro artefato
- Tipo do problema reportado
  - código
  - consulta
  - documentação
  - especificação (*design*)
  - sugestão

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

## Registro de problema



Laboratório de Engenharia de Software

- Severidade
  - Possíveis danos provocados pelo problema
  - É possível continuar a usar?
    - não, provoca danos sérios
    - não, é impossível utilizar os resultados
    - sim, se evitar a região problemática
    - sim, usando outra sequência de trabalho
    - sim, se desprezar alguns resultados
    - sim, pois somente incomoda
- É reproduzível?
  - identificação do problema
  - descrição do problema e como reproduzi-lo
- Sugestão de solução
  - isso nem sempre é desejável

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

## Registro de problema



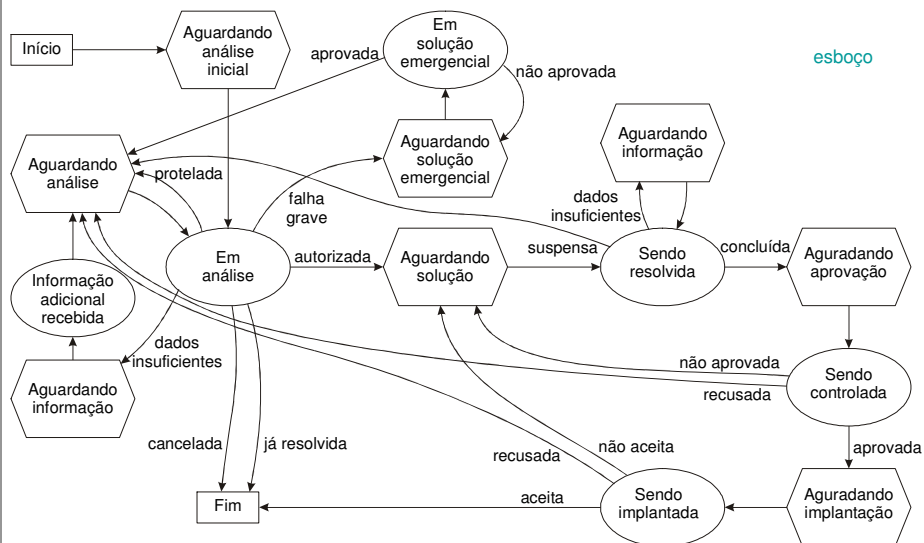
- Solução
  - estado da solução
    - datas de mudança de estado
    - responsáveis pelo trabalho nos estados de execução (ver a seguir)
  - descrição da solução
  - artefatos criados, alterados, eliminados
    - versões resultantes
  - possíveis causas das faltas identificadas
- FAP - Ficha de acompanhamento de problemas
  - registra o problema e a evolução da solução até ter sido completamente resolvido

Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

31

## Acompanhamento de problemas (*issue tracking*)



Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

32



## Técnicas de controle da qualidade



- Técnicas de controle **sem execução do artefato**
  - Prova formal da corretude
    - demonstração matemática da *correspondência exata* entre o artefato e a sua **especificação formal**
  - Argumentação da corretude
    - verificação baseada em matemática da correspondência entre o artefato e a sua especificação *suficientemente* formal
      - não necessariamente formal
    - utiliza os princípios de prova formal da corretude, mas sem o mesmo rigor
      - ex. assume-se que funções e/ou pseudo-instruções implementam corretamente a sua especificação
    - um programa argumentado correto *pode conter* defeitos
      - infelizmente a prática mostra o mesmo para programas provados corretos, embora com frequência menor

Yelowitz, L.; Gerhart, S.L.; "Observations of fallibility in applications of modern programming methodologies"; *IEEE Transactions on Software Engineering* 2(9); 1976; pages 195-207

Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

33

## Técnicas de controle da qualidade




- Técnicas de controle **sem execução do artefato**
  - Verificação de modelos
  - Revisões
    - leitura do artefato, com ou sem narrações para terceiros
  - Inspeções
    - semelhante a revisões, mas realizadas segundo um procedimento definido, documentado e controlado
  - Desenvolvimento em pares
    - duas pessoas trabalhando juntas em uma mesma estação de trabalho
    - uma digita e a outra controla o que está sendo digitado, dá sugestões, verifica a aderência a padrões, ...

Fev 2015

Arndt von Staa © LES/DI/PUC-Rio


34

Técnicas de controle da qualidade


Laboratório de Engenharia de Software

- Técnicas de controle **sem execução do artefato**
  - Teste estático, análise estática
    - exame de propriedades de um artefato sem pô-lo em operação
    - exemplos:
      - verificar se os padrões de programação estão sendo observados
      - verificar se as grandezas envolvidas no cálculo são coerentes
      - verificar se, para cada **throw**, existe um **catch** capaz de interceptar a exceção sinalizada
      - verificar se pode ocorrer *deadlock* ou condição de corrida
      - verificar se as assertivas são asseguradas pelo código
        - » possível só em parte
  - Medição estática
    - obtenção de medidas estruturais relativas ao artefato
      - as medidas indicam a probabilidade da presença de problemas  
→ *bad smells*
        - » ex. complexidade (número) ciclomática (McCabe) correlaciona (supostamente) com a densidade de defeitos

Fev 2015
Arndt von Staa © LES/DI/PUC-Rio
35

Técnicas de controle da qualidade



Laboratório de Engenharia de Software

- Técnicas de controle **com execução indireta**
  - Simulações
    - modelos que permitem prever ou avaliar propriedades do artefato (especificação)
  - Protótipos
    - versões experimentais e **descartáveis** de aspectos do artefato
      - não são liberações (*releases*) em um desenvolvimento incremental!

Fev 2015
Arndt von Staa © LES/DI/PUC-Rio
36

Laboratório de Engenharia de Software

## Técnicas de controle da qualidade



- Técnicas de controle **com execução direta**
  - Testes
    - condução de experimentos controlados envolvendo a execução do artefato
  - Medição dinâmica
    - obtenção de medidas relativas ao comportamento do artefato durante a execução
  - Instrumentação
    - código de controle da integridade ou de medição contido nos artefatos
    - código de controle da cobertura dos testes contido nos artefatos
  - Aprovação a cada iteração
    - teste realizado pelo usuário a fim de verificar se o construto corresponde às suas expectativas explícitas e implícitas
    - viabiliza o controle da qualidade de especificações antes de se dispor do sistema completo

Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

37

Laboratório de Engenharia de Software

## Referências



- Borba, P.; Cavalcanti, A.; Sampaio, A.; Woodcock, J.; eds.; *Testing Techniques in Software Engineering*; Berlin: Springer, Lecture Notes in Computer Science; LNCS 6153; 2010
- Staa, A.v.; *Programação Modular*; Rio de Janeiro: Campus; 2000
- Weinberg, G.M.; *The Psychology of Computer Programming*; 2nd edition; Dorset House; 1998
- Yelowitz, L.; Gerhart, S.L.; "Observations of fallibility in applications of modern programming methodologies"; *IEEE Transactions on Software Engineering* 2(9); 1976; pages 195-207

Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

38

Laboratório de Engenharia de Software

LES

FIM

Fev 2015

Arndt von Staa © LES/DI/PUC-Rio

39