


Laboratório de Engenharia de Software

Garantia da qualidade e Técnicas de controle da qualidade

Arndt von Staa
Departamento de Informática
PUC-Rio
Fevereiro 2017

Especificação



Laboratório de Engenharia de Software

- Objetivo da aula
 - Conceituar garantia da qualidade
 - 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 2017Arndt von Staa © LES/DI/PUC-Rio2

Laboratório de Engenharia de Software

Garantia da qualidade

- É o conjunto de **atividades sistemáticas** visando **assegurar a adequação ao uso** do artefato como um todo
 - torna imprescindível
 - existir e obedecer a um processo de trabalho definido
 - existir um ambiente coerente com o processo e que assegure ganhos de produtividade e continuidade do trabalho
 - ferramentas interdependentes
 - existir uma especificação fidedigna
 - realizar “continuamente” controle da qualidade
 - realizar medições (introspecção) como instrumento de aprendizado com vistas à melhora do processo

adequação ao uso: é (muito) alta a **probabilidade** de que o software satisfaça plenamente as necessidades e anseios explícitos e implícitos do usuário e do cliente (fidedignidade)



Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
3

Laboratório de Engenharia de Software

Processo MPS para garantia da qualidade

9.2.3 Processo: Garantia da Qualidade - GQA
Nível MR-MPS: F - Gerenciado

Propósito:
O propósito do processo Garantia da Qualidade é assegurar que os produtos de trabalho e a execução dos processos estejam em conformidade com os planos, procedimentos e padrões estabelecidos.


Resultados esperados:

- GQA 1. A aderência dos produtos de trabalho aos padrões, procedimentos e requisitos aplicáveis é avaliada objetivamente, antes dos produtos serem entregues e em marcos predefinidos ao longo do ciclo de vida do projeto;
- GQA 2. A aderência dos processos executados às descrições de processo, padrões e procedimentos é avaliada objetivamente;
- GQA 3. Os problemas e as não-conformidades são identificados, registrados e comunicados;
- GQA 4. Ações corretivas para as não-conformidades são estabelecidas e acompanhadas até as suas efetivas conclusões. Quando necessário, o escalamento das ações corretivas para níveis superiores é realizado, de forma a garantir sua solução;

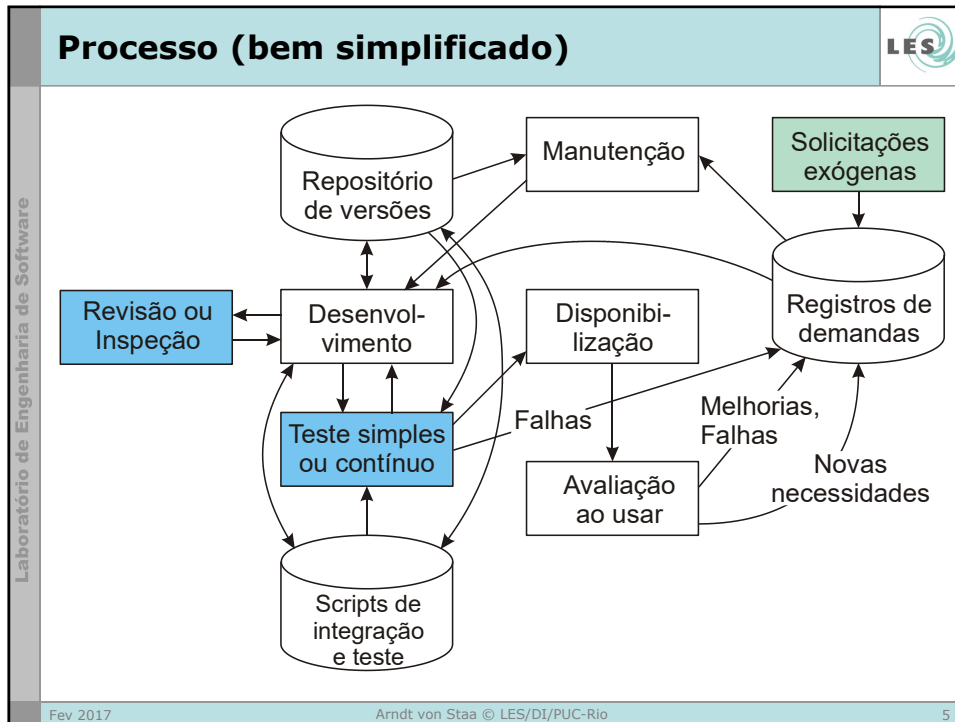
Informações adicionais para definição e implementação do processo:

- Consulte NBR **ISO/IEC 12207** e as emendas 1 e 2 da ISO/IEC 12207: Subprocesso Garantia da Qualidade
- Consulte **ISO/IEC 15504-5**: Processo Garantia da Qualidade
- Consulte **CMMI-SE/SW**: Área Garantia da Qualidade

http://www.softex.br/wp-content/uploads/2016/04/MPS.BR_Guia_Geral_Software_2016-com-ISBN.pdf



Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
4



Solicitações exógenas

- Solicitações exógenas são tipicamente especificações:
 - requisitos funcionais, não funcionais ou inversos
 - solicitações de mudanças
 - relatos de falhas
 - solicitações de evolução, adaptação
 - ...
- São redigidos por
 - interessados
 - desenvolvedores
 - mantenedores
 - ...
- Podem estar redigidos nos mais variados níveis de abstração
- Podem estar incompletas e/ou imprecisas, ex. relatos de falha
- As solicitações são registradas na base dados de demandas (*issue tracker*)

Fev 2017 Arndt von Staa © LES/DI/PUC-Rio 6

Especificação de software



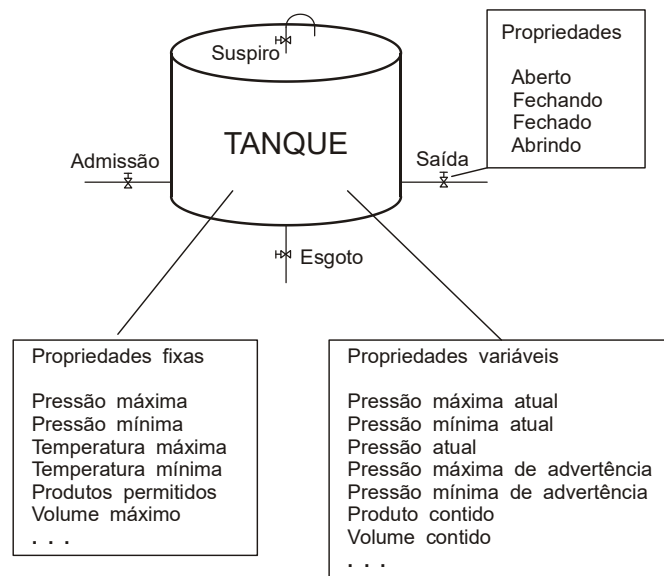
- Especificações de software são tipicamente **representações**, escritas em alguma **linguagem de representação**, formada por textos, tabelas e/ou diagramas
 - muitas dessas linguagens são *ad hoc*, ou seja inventadas pelo redator e não possuem sintaxe nem semântica definida
- Especificações são frequentemente extensas, confusas, ambíguas, incompletas, incorretas, não verificáveis, ...
 - portanto são fontes de defeitos a serem propagados
- Consequência: são frequentes as especificações erradas
 - um grande número dos defeitos encontrados em sistemas em operação são defeitos de especificação

Fev 2017

Arndt von Staa © LES/DI/PUC-Rio

7

Ex. linguagem de uma representação ad hoc




Fev 2017

Arndt von Staa © LES/DI/PUC-Rio

8

Laboratório de Engenharia de Software

Propriedades básicas de representações




- Representações servem de **mecanismo de comunicação** entre todos os interessados
- **Consequência**: todas as representações precisam satisfazer os seguintes requisitos de qualidade:
 - **inteligibilidade** – todos interessados entendem a representação
 - **vocabulário usado** – deve ser compreensível a todos os leitores e respeitado por todos
 - dicionário de termos / dicionário de dados
 - cada **termo** tem um significado único e bem definido
 - cada **significado** corresponde a um termo único e bem definido
 - **sintaxe** – a representação estar sintaticamente correta com relação à linguagem de representação usada
 - **semântica** – todos os interessados entendem e concordam que a representação seja satisfatória para os seus interesses

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
9

Laboratório de Engenharia de Software

Derivação, terminologia



- Especificação **antecedente** é a especificação do artefato a desenvolver
- Artefato **consequente** é um artefato derivado (transformado) a partir de uma especificação antecedente
- O artefato consequente pode ser
 - uma especificação em **nível de abstração mais baixo**
 - um artefato “folha”, i.e. do qual não se deriva mais nada
- Uma especificação antecedente pode derivar vários artefatos consequentes
 - o conjunto de artefatos consequentes é interdependente

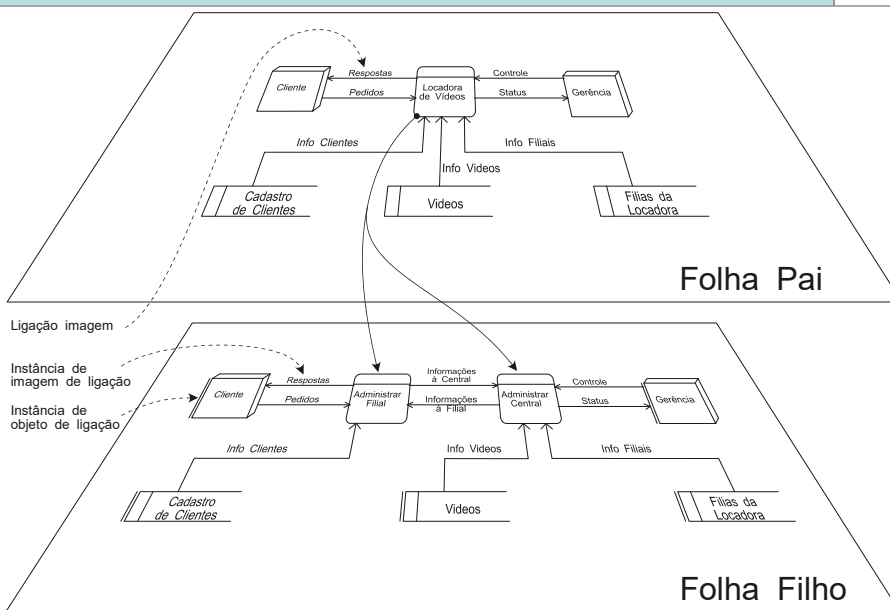
Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
10

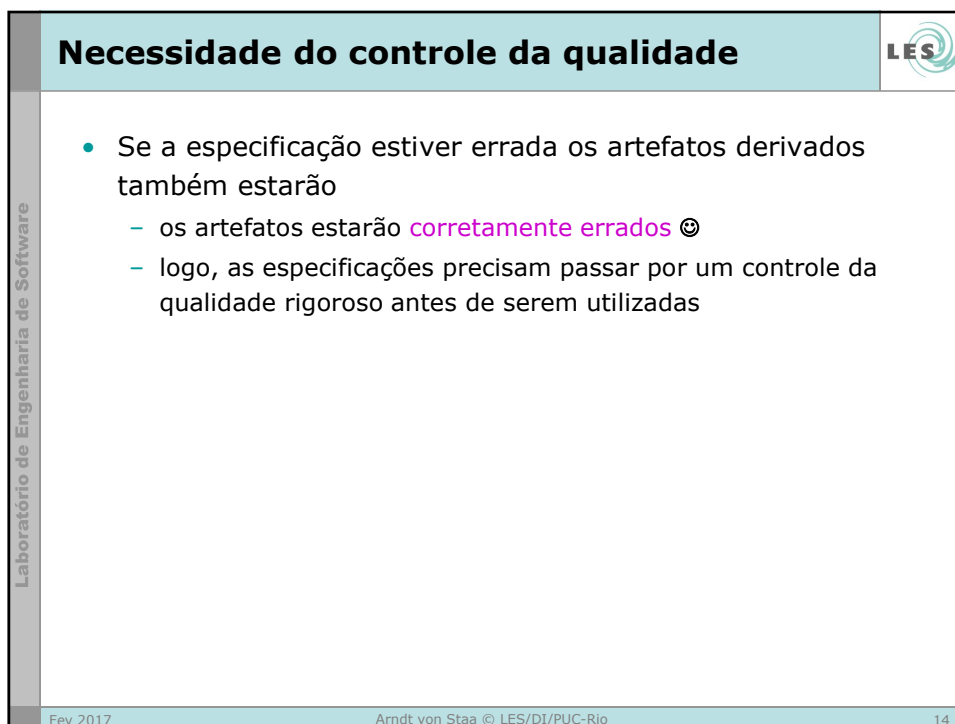
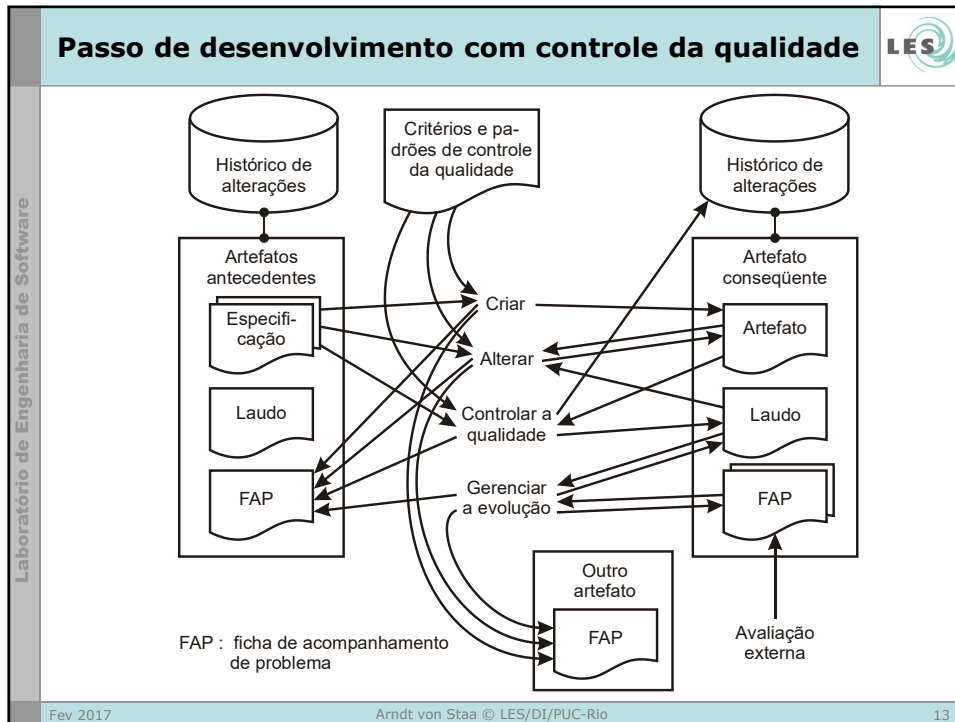
Objetivo do controle da qualidade



- Verificar se especificações estão completas e corretas
- Verificar se os **artefatos consequentes** de uma especificação estão em **correspondência exata** com esta especificação
- Verificar se as **interdependências** existentes entre os artefatos consequentes estão definidas, corretas e completas


Derivação, exemplo





Laboratório de Engenharia de Software

Necessidade do controle da qualidade




- Para que se possa dizer que o artefato derivado está correto tudo que for especificado precisa ser **observável**
 - quanto maior a **formalidade**
 - mais observável será
 - menos defeitos conterà
 - **problema**: poucos sabem ler e compreender especificações formais, ou quase formalizadas

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
15

Laboratório de Engenharia de Software

Especificação inicial




- A **especificação inicial** obviamente não é derivada de outra especificação
- A **especificação inicial** contém os desejos explicitados pelos interessados
 - deveria conter todos os requisitos
 - nada do que não está especificado deveria ser considerado óbvio ou de conhecimento geral
- Também é especificação inicial qualquer adição a um artefato consequente e que não pode ser obtida por simples transformação da especificação antecedente

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
16

Laboratório de Engenharia de Software

Exemplo de adição de conhecimento

- Uma especificação de arquitetura precisa estar em **conformidade** com a especificação de requisitos inicial
- Porém contém um volume considerável de **requisitos técnicos** que visam tornar possível satisfazer a especificação de requisitos dos interessados
 - **reificação** – tornar real, ou menos abstrata, uma abstração
- Os requisitos técnicos são adicionados pelos arquitetos
 - precisam
 - ser compreensíveis pelos desenvolvedores ☹
 - estar corretos, completos, ...
 - ser coerentes com toda a especificação inicial




Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
17

Laboratório de Engenharia de Software

Especificação inicial


- O controle da qualidade de uma especificação inicial depende da **precisão da comunicação** entre os interessados e os desenvolvedores
- Muitos interessados são **leigos** em computação
 - podem não entender a linguagem de representação usada,
 - podem não entender a necessidade da precisão de especificação
 - ruim é quando desenvolvedores também não entendem isso...
 - ao iniciar nem sempre têm um conhecimento completo, correto e viável do que desejam



Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
18

Laboratório de Engenharia de Software

Especificação através de exemplos




Com o intuito de reduzir os problemas de comunicação

- Que tal redigir um conjunto de **cenários de teste**
 - em conformidade com **o que se quer** especificar **antes de iniciar** o desenvolvimento dos artefatos dependentes
 - os cenários são redigidos em linguagem natural simplificada
 - os cenários são essencialmente **exemplos de uso**
- Os exemplos de uso
 - **permitem** aos interessados dizer se a especificação corresponde aos desejos explícitos e (alguns) implícitos
 - **ajudam** o desenvolvedor a entender o que é para ser feito
- Portanto,
 - são **instrumentos eficazes para reduzir o retrabalho inútil**

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
19

Laboratório de Engenharia de Software

Exemplo




- Como interessado em me cadastrar para usar o sistema X
 - quando ativar o sistema X
 - quando solicitar a função cadastrar
 - quando tiver preenchido todos os campos obrigatórios do formulário de cadastramento
 - quando tiver digitado exatamente o captcha
 - quando clicar a ação "Inserir"
 - então o sistema X me incluirá em seu cadastro
- Incompleto?
 - qual o conteúdo do formulário?
 - que caracteres são permitidos em cada um dos campos?
 - o que é um captcha?

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
20

Laboratório de Engenharia de Software

Exemplo: dicionário de dados / termos




- Registro de cadastro do interessado
 - endereço e-mail do interessado
 - um ou mais telefones, no máximo 5, onde possa ser encontrado
 - identificação de usuário
 - a identificação deve ser única segundo o cadastro de usuários
 - senha
 - a força da senha é controlada segundo regras que me podem ser informadas
- Captcha – é uma imagem formada por caracteres redigidos de forma que não sejam reconhecíveis por um OCR – optical character recognition. Usa-se captchas em formulários de entrada para evitar que robôs façam uso do sistema.

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
21

Laboratório de Engenharia de Software


Exemplo: regras de negócio



- Nome do usuário
 - valem letras maiúsculas, minúsculas e diacríticos
 - valem os caracteres `-'`.' e branco
- Identificação do usuário
 - cada identificação contida no cadastro deve se referenciar a um único usuário autorizado
 - valem letras maiúsculas e minúsculas
 - não valem diacríticos
 - valem os caracteres `-' e `.'
 - não valem brancos
- Senha
 - valem letras maiúsculas e minúsculas
 - valem os caracteres especiais: !@#\$%&*()_+={ }[,].,:;
 - deve ser forte segundo regras informadas ao usuário
 - ao definir deve ser digitada, não pode usar cortar e colar

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
22

Processo visando qualidade e economia




Laboratório de Engenharia de Software

- Todos os artefatos devem ser desenvolvidos procurando, desde o início de seu desenvolvimento, minimizar a possível existência de defeitos
 - quanto menos defeitos existirem antes de se realizar os primeiros testes, mais nos aproximamos do ideal **correto por construção**
 - quanto mais cedo **defeitos acidentalmente injetados** forem identificados, menor será o **retrabalho inútil**
 - quanto mais **defeitos remanescentes** forem identificados e removidos **antes de liberar** para o uso, mais nos aproximamos do ideal **correto por desenvolvimento**
 - isso usualmente implica na revisão de artefatos antecedentes
 - quanto **menos recursos** forem necessários para controlar a qualidade e corrigir, mais **econômico** será o desenvolvimento

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
23

Objetivos do controle



Laboratório de Engenharia de Software


- O objetivo do controle da qualidade é **identificar defeitos** (discrepâncias) com relação a
 - *interesses explícitos e implícitos dos interessados*
 - requisitos (especificações) *funcionais, não funcionais, inversos, de contrato*
 - requisitos de **interface humano-sistema**
- Assegurando **baixo custo** considerando toda a vida útil
 - CTP – *custo total de propriedade* (TCO – *total cost of ownership*)
 - também envolve custos de manutenção, operação e uso

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

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
24

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**
 - **problema da existência**: podemos procurar e encontrar defeitos, mas se não encontrarmos **não podemos concluir que não existam**


- estado da prática
- estado da arte

- o que se **pratica costumeiramente** nas organizações
- o nível de desenvolvimento mais avançado que se consegue com o **conhecimento atual**

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
25

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 2017
Arndt von Staa © LES/DI/PUC-Rio
26

Laboratório de Engenharia de Software

Propriedades do controle: assegura o que?




- Controle da qualidade **não assegura** qualidade!
 - controle da qualidade **não corrige!**
 - o resultado do controle é **meramente um laudo**
 - controle da qualidade somente verifica o **quanto o artefato se afasta da qualidade** desejada
 - procura **encontrar e relatar** defeitos
 - **diretamente**
 - ou **indiretamente** a partir de falhas que permitam determinar os defeitos causadores
- Porém, saber como será controlado **antes de desenvolver** induz o desenvolvedor a se **aproximar, por construção, da qualidade requerida**

• 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 2017
Arndt von Staa © LES/DI/PUC-Rio
27

Laboratório de Engenharia de Software

Propriedades do controle: planejamento?




- A forma de realizar o controle da qualidade de cada artefato deve ser planejada (definida) **junto com a correspondente 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 2017
Arndt von Staa © LES/DI/PUC-Rio
28

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 deveriam ser **testáveis**
 - de preferência de forma automática

Laboratório de Engenharia de Software

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
29

Laboratório de Engenharia de Software

Propriedades do controle



- Após cada alteração é necessário repetir o controle da qualidade
 - **teste de regressão**
 - verificar se tudo que não foi afetado pela alteração continua funcionando corretamente
 - isso compromete a produtividade
 - gera perdas devido ao retrabalho
 - retrabalho inútil, quando se trata de correção


Custos? Como evitar que cresçam demasiadamente?

Laboratório de Engenharia de Software

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
30

Laboratório de Engenharia de Software

Propriedades do controle, redução do custo



- Sugestão: **automatizar** o controle da qualidade
 - como fazê-lo com relação a código?
 - é possível ser feito para tudo 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 2017

Arndt von Staa © LES/DI/PUC-Rio

31

Laboratório de Engenharia de Software

Tarefas do controle da qualidade



- **Verificação**: controle da qualidade de um **artefato isolado** com relação à sua especificação e aos padrões exigidos
 - examina se o artefato está em **conformidade** exata com a sua especificação
 - examina a **conformidade** com os padrões exigidos
 - ex. examina o **correto uso das linguagens** de representação
 - examina se o **artefato** forma um todo **coerente, coeso e completo**
 - examina se a **intenção do redator** é correta e completamente compreendida ao ler as representações que constituem o artefato
 - não contém subentendidos
 - não está fora de foco
 - nada falta
 - não contém excesso
 - não contém ambiguidades
 - examina se está em um **nível uniforme de abstração**
 - ex. especificação não deve conter código

exato – nem mais, nem menos


Fev 2017

Arndt von Staa © LES/DI/PUC-Rio

32

Laboratório de Engenharia de Software

Tarefas do controle da qualidade




- **Validação**: controle da qualidade de uma conjunto de artefatos que formam um **construto entregável** ao usuário com relação à especificação do construto e respectivos padrões exigidos
 - examina se não existem conflitos entre artefatos
 - em especial entre **visões**
 - estrutural
 - funcional
 - dinâmica
 - usuário
 - examina se o **conjunto de artefatos forma um todo coerente, coeso e completo**
 - ex. o conjunto de módulos implementa uma determinada característica do sistema?
 - examina se todas as **interfaces entre artefatos são respeitadas**

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
33

Laboratório de Engenharia de Software

Tarefas 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**

existe corretude absoluta, i.e. segundo qualquer possível forma de controle utilizada e independente do observador?

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
34

Laboratório de Engenharia de Software

Tarefas do controle da qualidade




- **Aceitação**: aceitação controle da qualidade de um **construto entregável** ao usuário com relação às **atuais** necessidades e expectativas **explícitas** e **implícitas** dos seus interessados
 - para **antecipar a defeitos encontrados durante a aceitação** é recomendado desenvolver de forma **incremental**
 - cada novo construto adiciona poucas características com relação ao anterior
 - ao término de cada incremento examina-se a satisfação das necessidades explícitas e implícitas dos usuários
 - reduz o espalhamento de defeitos devido a erros de especificação
 - implica que cada incremento leve a alguma coisa útil

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
35

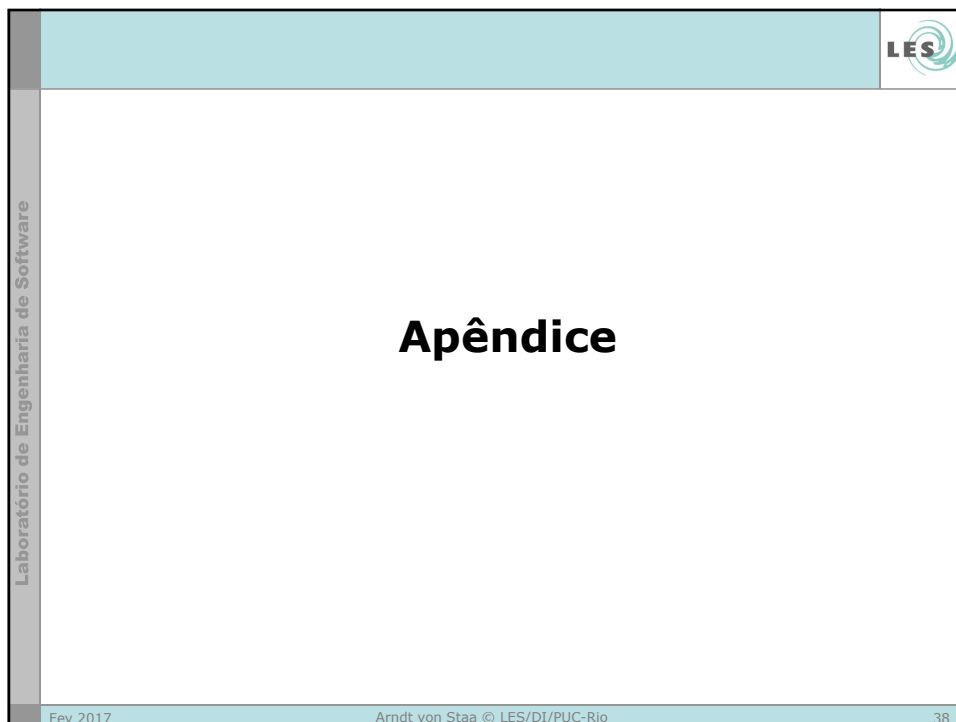
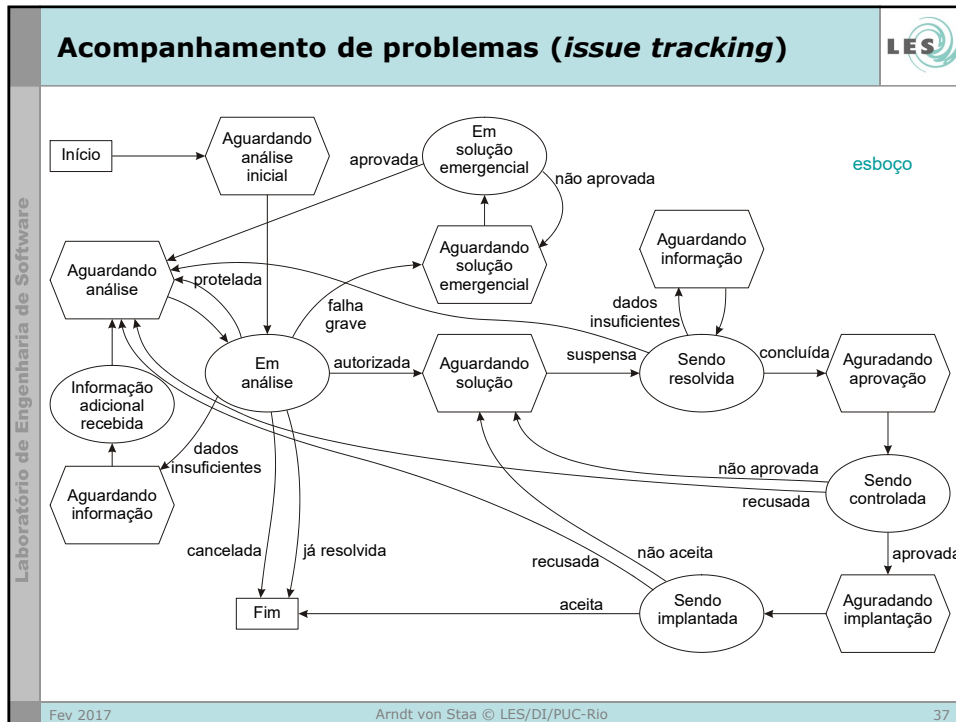
Laboratório de Engenharia de Software

Tarefas do controle da qualidade




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

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
36



Garantia da qualidade: atividades 1/4




Laboratório de Engenharia de Software

- Verificar se **são obedecidas** todas as normas, padrões, práticas e convenções estipuladas
- Verificar se as **documentações técnica** e de **uso**
 - são produzidas
 - são utilizadas para desenvolver
 - são mantidas atualizadas – **co-evoluídas**
 - estão sempre disponíveis
 - são inteligíveis por todos os leitores a que se destinam
 - contêm tudo o que deveriam conter

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
39

Garantia da qualidade: atividades 2/4




Laboratório de Engenharia de Software

- Verificar se o **controle da qualidade** é realizado conforme determinado
 - critérios de aceitação devem estar estabelecidos previamente
- Verificar se os problemas identificados pelo controle da qualidade são **registrados** e **acompanhados** até a sua completa resolução
 - acompanhamento de demandas (*issue tracking*)
 - gerência da configuração
 - gerência de requisitos
 - gerência da evolução e da adaptação

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
40

Laboratório de Engenharia de Software

Garantia da qualidade: atividades 3/4




- Verificar se **medições** são coletadas e **usadas para melhorar**:
 - o processo de desenvolvimento
 - padrões de uso das linguagens de representação
 - qualidade da engenharia
- Verificar se métodos, técnicas e ferramentas são **adequadas e utilizadas** por todos os participantes
- Verificar se todos utilizam as **mesmas versões das ferramentas** e os mesmos parâmetros de configuração
- Verificar se os artefatos aprovados são **armazenados em bibliotecas controladas**
 - controle de versões

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
41

Laboratório de Engenharia de Software


Garantia da qualidade: atividades 4/4



- Verificar se existe um sistema de **backup** adequado
 - assegurar que este sistema funciona corretamente
- Verificar se o **peçoal** envolvido no projeto
 - é **disciplinado**
 - possui **proficiência** suficiente
 - está **habilitado** a usar as ferramentas, padrões e processos disponíveis
 - dispõe de oportunidades para obter o **treinamento necessário**
 - evolução profissional contínua

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
42

Técnicas de controle da qualidade




Laboratório de Engenharia de Software

- Técnicas de controle **sem execução do artefato**
 - Verificação do dicionário de dados (vocabulário permitido)
 - Verificação da sintaxe e da semântica estática
 - Verificação dos 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 2017
Arndt von Staa © LES/DI/PUC-Rio
43

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 2017
Arndt von Staa © LES/DI/PUC-Rio
44

Laboratório de Engenharia de Software

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; pags 195-207

Fev 2017
Arndt von Staa © LES/DI/PUC-Rio
45

Laboratório de Engenharia de Software


Técnicas de controle da qualidade



- 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 2017
Arndt von Staa © LES/DI/PUC-Rio
46

Técnicas de controle da qualidade




Laboratório de Engenharia de Software

- 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 2017
Arndt von Staa © LES/DI/PUC-Rio
47

Laudo, registro de problemas




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 2017
Arndt von Staa © LES/DI/PUC-Rio
48

Laboratório de Engenharia de Software

Laudo, registro de problemas



- 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 reprodutível?
 - identificação do problema
 - descrição do problema e como reproduzi-lo
- Sugestão de solução
 - isso nem sempre é desejável


Fev 2017

Arndt von Staa © LES/DI/PUC-Rio

49

Laboratório de Engenharia de Software

Laudo, registro de problemas



- 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 2017

Arndt von Staa © LES/DI/PUC-Rio

50

Laboratório de Engenharia de Software

Referências




- Adzic, G.; *Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing*; London, UK: Neuri, Kindle edition; 2009
- Borba, P.; Cavalcanti, A.; Sampaio, A.; Woodcock, J.; eds.; *Testing Techniques in Software Engineering*; Berlin: Springer, Lecture Notes in Computer Science; LNCS 6153; 2010
- Charette, R.N.; "Why software fails"; *IEEE Spectrum* 42(9); IEEE Computer Society; 2005; pags 42-49
- Jones, C.; Bonsignour, O.; *The Economics of Software Quality*; Upper Saddle River, NJ: Addison Wesley, Kindle edition; 2012
- Ross, P.E.; "The Exterminators"; *IEEE Spectrum* 42(9); IEEE Computer Society; 2005; pags 36-41
- Staa, A.v.; *Programação Modular*; Rio de Janeiro: Campus; 2000
- Weinberg, G.M.; *The Psychology of Computer Programming*; 2nd edition; Dorset House; 1998

Fev 2017

Arndt von Staa © LES/DI/PUC-Rio

51

Laboratório de Engenharia de Software



FIM

Fev 2017

Arndt von Staa © LES/DI/PUC-Rio

52