


Laboratório de Engenharia de Software

Revisões e Inspeções 1

Arndt von Staa
Departamento de Informática
PUC-Rio
Março 2015

Especificação




Laboratório de Engenharia de Software

- Objetivo da aula
 - apresentar técnicas de revisão, inspeção e leitura
- Justificativa
 - O controle da qualidade deve ser realizado continuamente
 - junto com o desenvolvimento
 - ainda **antes de se dispor do código**.
 - Algumas propriedades do código são **difíceis ou muito caras de testar**.
 - torna-se necessário o uso de técnicas de **controle da qualidade alternativas aos testes**.
 - **Revisões, inspeções e leitura dirigida**, se bem realizadas, têm mostrado muito bons resultados.
- Pezzè, M.; Young, M.; *Teste e Análise de Software*; 2008; capítulo 18
- Staa, A.v.; *Programação Modular*; Campus; 2000; capítulos 3, 10 e 12

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

Artefato, conceitos




Laboratório de Engenharia de Software

- **Artefato** (*work product*):
 - é qualquer **resultado tangível** do desenvolvimento
 - está escrito, ou desenhado, e armazenado
 - e que teve a sua **qualidade controlada** de alguma forma

Mar 2015Arndt von Staa © LES/DI/PUC-Rio3

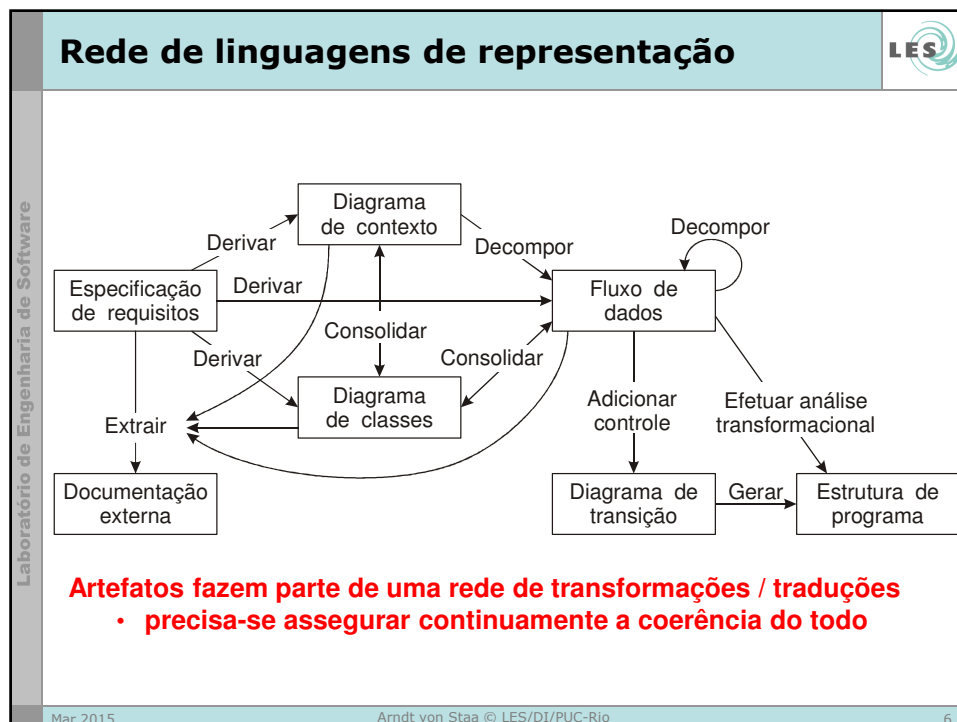
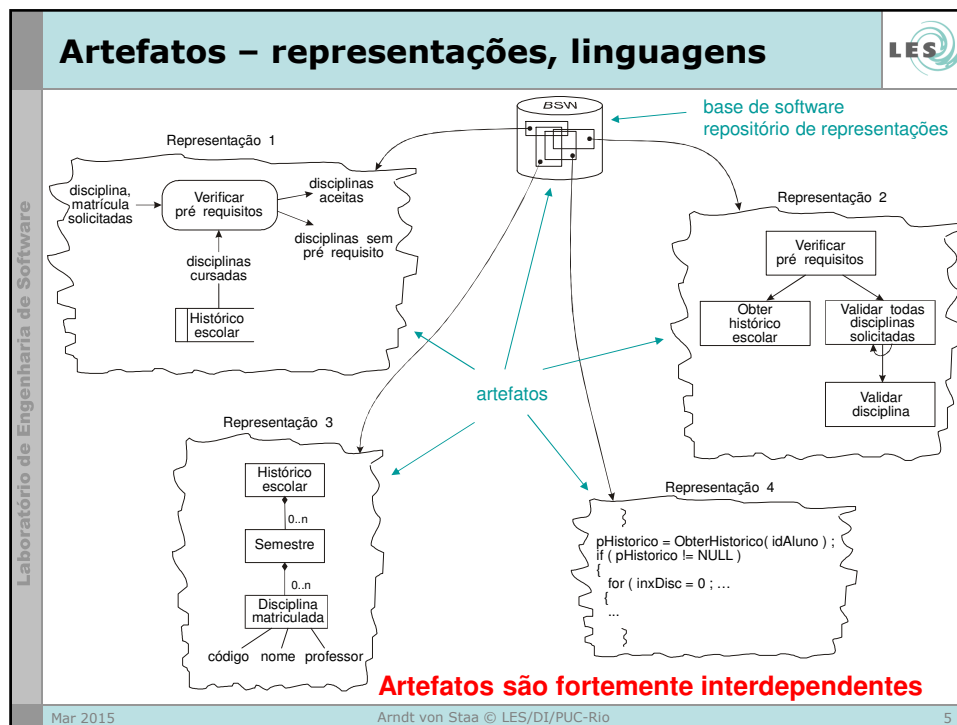
Artefato, conceitos



Laboratório de Engenharia de Software

- Cada artefato é composto por uma ou mais **representações**, cada qual escrita em uma **linguagem de representação**
- Linguagens de representação
 - podem ser
 - textuais
 - formulários ou tabulares
 - gráficas
 - possuem
 - sintaxe
 - semântica
 - nível de abstração

Mar 2015Arndt von Staa © LES/DI/PUC-Rio4



Propriedades de artefatos **anteriores**



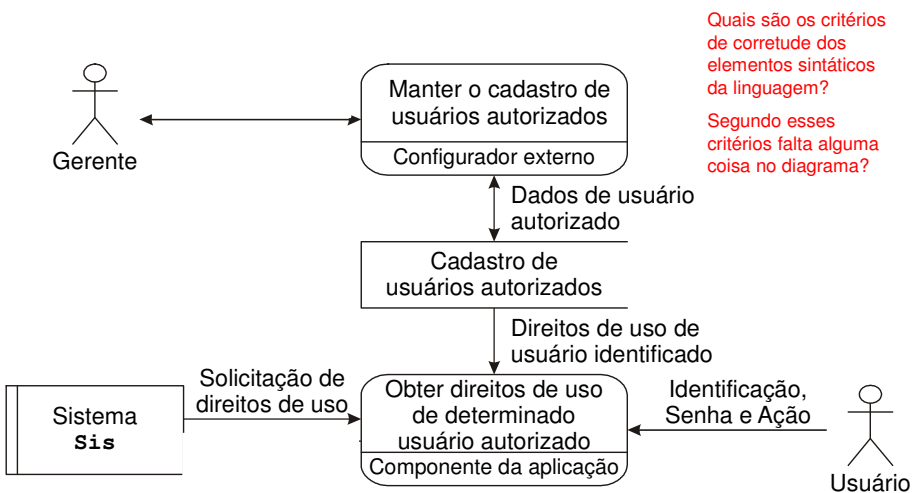
- Uma especificação, arquitetura, ou projeto deve:
 - deixar claro o que deve ser implementado
 - apoiar o controle da qualidade
 - do próprio artefato, ex.
 - sintaxe
 - semântica
 - padrões de uso
 - do artefato consequente, ex.
 - rastreamento da transformação
 - do resultado da implementação, ex.
 - através de suítes de teste derivadas a partir do artefato antecedente

Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

7

Como verificar completude e corretude?




Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

8

Principais tipos de defeitos em **projetos**




Laboratório de Engenharia de Software

Defeito	Descrição	Aplicado a projeto
Omissão	falta informação necessária no artefato	um ou mais requisitos ou características não são abordados no projeto
Incorreção	alguma informação contida no artefato conflita com o domínio do problema, ou com o serviço a ser prestado	o projeto contém uma reificação errônea de um requisito
Inconsistência	alguma informação contida no artefato contradiz o que se encontra em outros artefatos ou mesmo neste artefato	um conceito registrado no projeto está em desacordo com o que se encontra em outros artefatos

Reificação: No processo de alienação, o momento em que a característica de ser uma "coisa" se torna típica da realidade objetiva. [Aurélio eletrônico] → determina como resolver um requisito ou uma abstração

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

Principais tipos de defeitos em projeto



Laboratório de Engenharia de Software

Defeito	Descrição	Aplicado a projeto
Ambiguidade	Informação contida no artefato admite uma variedade de interpretações	Um conceito contido no projeto favorece dúvidas quanto ao seu significado, possibilitando um erro de compreensão
Desnecessário	Informação contida no artefato não é utilizada, ou pertence a outro nível de abstração	O projeto contém informação que, mesmo se relevante, não deveria se encontrar nesse artefato
Duplicata	Um mesmo conceito é definido ou especificado em dois ou mais artefatos ou em diferentes locais de um mesmo artefato	O projeto especifica de novo um conceito já existente em um ou mais outros artefatos risco: inconsistência

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

Como verificar o texto a seguir?



Descrição do módulo `RDT_ReadTestScript` do arcabouço de teste

Provê funções para a leitura e análise léxica dos comandos de teste.

Pressupõe-se que cada comando de teste esteja integralmente em uma linha.

Cada comando de teste inicia com o caractere '=' seguido de um string que identifica o comando.

Cada comando pode requerer zero ou mais parâmetros que se encontram na mesma linha que o comando.

Parâmetros podem ser literais ou simbólicos.

Os parâmetros simbólicos precisam estar declarados antes de serem utilizados.

Os parâmetros têm tipo e a leitura deve respeitar esses tipos.

Podem existir linhas estabelecendo um "#include"

Se for do interesse do programador, módulos de teste específico podem ler e processar por conta própria linhas do script. Isto pode ser necessário quando um módulo necessita de um grande número de parâmetros ou dados especiais.

Adaptado do arcabouço de teste C++, módulo `ReadTest`

Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

11

Como verificar a interface do módulo



```
RDT_ReadTestScript( SMT_SymbolTable * pTable )
~RDT_ReadTestScript( )
RDT_tpRetCode OpenTestScriptFile( char * FileNameParm )
STR_String * GetTestScriptFileName( )
int GetNumberLinesRead( )
int ReadTestScriptLine( )
int ReadCommandLine( char * fieldTypes , ... )
bool ReadCommand( char * Command , int dimCommand )
bool ReadName( int * sizName , char * Name , int dimName )
bool ReadChar( char * Char )
bool ReadDouble( double * Double )
bool ReadInt( long * Int )
bool ReadBool( bool * Bool )
bool ReadString( int * sizString , char * String , int dimString )
```

Isso complementa ou substitui o slide anterior?

Obs: ... número variável de parâmetros

- Basta isso para saber como implementar esse módulo?
- Basta isso para saber como testar o módulo?
- Como você testaria usando um script interpretado pelo módulo?

Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

12

Especificação detalhada: ReadCommandLine



```
// Description
// Decodes the line that is containing in the read buffer, skipping the command.
//
// Parameters
// $P TypeList is a string identifying the types of the parameters. Types are defined by a single letter:
//      b - bool, literals are .f or .false; .t or .true
//      c - char
//      d - double
//      i - integer
//      s - string - require two receiving parameters the length and the string itself
// $P ... references to the spaces where the values should be stored
//
// As with scanf the lists of types and parameters must be consistent both in length and type
// When reading a field, first is verified whether it is a name that corresponds to a declared parameter
// If not, the value read is treated as a literal.
// The syntax of names and literals is identical to the syntax of C/C++ program names and literals.
// Errors are issued if the syntax is incorrect, if parameters are undefined, or if types do not match.
// Strings may contain any character, including zeroes and blanks.
// CAUTION:
// A string is ONE field, although it returns two parameters.
//
// Return value
// Returns the number of fields read until end of the type list or
// until an error has been found.
...
```

Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

13

Revisões e inspeções: definições



- Uma **revisão** é uma **leitura crítica** do artefato e da documentação complementar, **anotando**:
 - os **defeitos** encontrados
 - as **dúvidas** e dificuldades de compreensão
 - as possibilidades de **melhoria**
- Uma **inspeção** é uma **leitura crítica formalizada** do artefato e da documentação complementar
 - segundo um **plano** definido
 - obedecendo a **regras de leitura** definidas
 - envolvendo uma **pequena equipe**
 - anotando **defeitos, dúvidas e melhorias**


Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

14

Laboratório de Engenharia de Software

Revisões e inspeções: aplicam-se a o que?




- Revisões e inspeções podem ser utilizadas para controlar a qualidade de **qualquer artefato** em **qualquer linguagem de representação** destinada a humanos
 - especificações
 - modelos
 - projetos
 - código
 - scripts de teste
 - processos de desenvolvimento definidos
 - planos
 - padrões
 - auxílio (help) para o usuário
 - documentação para o usuário
 - teses, dissertações, trabalhos de fim de curso
 - . . .

Mar 2015Arndt von Staa © LES/DI/PUC-Rio15

Laboratório de Engenharia de Software

Revisões e inspeções são eficientes



- Revisões e inspeções informam **diretamente** o defeito
 - em geral o custo é baixo para **localizar completamente** o defeito
 - podem ser realizadas com relação a artefatos não executáveis ou ainda incompletos

Mar 2015Arndt von Staa © LES/DI/PUC-Rio16

Revisões e inspeções vs. testes



- Contrastando, testes informam o sintoma (falha) obrigando diagnosticar a causa (defeito)
 - custo alto para **identificar a causa exata** e **localizar correta e completamente** o defeito
 - testes somente podem ser realizados com relação a artefatos executáveis
 - teste ocorre tarde no desenvolvimento
 - depois de já se ter sido comprometido muitos recursos

Revisões e inspeções vs. testes



- Revisões e inspeções são complementares a testes?
 - eliminam defeitos antes que se propaguem para outros artefatos ou para o serviço
 - um **defeito na especificação** provoca um **erro** ao criar a arquitetura (ou projeto), que se manifesta sob a forma de um **defeito na arquitetura**
 - um **defeito na arquitetura** provoca um **erro** ao criar o projeto, que se manifesta sob a forma de um **defeito no projeto**
 - um **defeito em um projeto** provoca um **erro** ao redigir o código, que se manifesta sob a forma de um **defeito no código** ou no **tratamento da interface**
 - a execução de um **defeito no código** causa um **erro endógeno**
 - lembre-se o erro **precisa ser observado** para tornar-se uma falha
 - um defeito no **tratamento de uma interface** pode estabelecer uma **vulnerabilidade** que, se explorada com sucesso, causará um **erro exógeno**

Laboratório de Engenharia de Software

Revisões e inspeções vs. testes




- Revisões e inspeções são complementares a testes?
 - revisões **devem** ser praticadas **a partir do primeiro momento** do desenvolvimento
 - **reduzem** significativamente o **retrabalho inútil**
 - quanto maior a **latência**, maior a poluição provocada pelo erro e mais cara e demorada fica a depuração

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

Laboratório de Engenharia de Software

Revisões e inspeções são eficazes




- Parcela significativa dos defeitos existentes em um artefato é encontrada através de revisões ou inspeções
 - segundo vários autores a inspeção, quando for praticada, encontra de **60% a 80% do total** dos defeitos encontrados
 - não é de se esperar?
 - se o controle é feito antes dos testes, então sobram menos defeitos a serem encontrados através dos testes
 - também sobram menos defeitos **remanescentes** entregues ao usuário
 - alguns autores mencionam que se **economiza perto de 40% do custo total** de desenvolvimento quando se praticam inspeções
 - não é de se esperar?
 - afinal a inspeção reduz o custo decorrente do retrabalho inútil

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

Laboratório de Engenharia de Software

Técnicas de revisar ou inspecionar



- **Leitura simples**
 - rever antes de prosseguir
 - habitue-se a **fazer isso sempre**
 - procure sempre ser rigoroso ao rever
 - **ler para encontrar defeitos**
- **Leitura dirigida segundo critérios estabelecidos**
 - padrões de uso das linguagens de representação
 - manual de critérios
 - pode envolver técnicas formais
 - argumentação, prova da correitude
- **Leitura baseada em pontos de vista**
 - segundo papéis desempenhados ou simulados pelo revisor

• Correção de trabalhos é um exemplo de revisão?


Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

21

Laboratório de Engenharia de Software

Leitura dirigida



- **Leitura baseada em pontos de vista** encontram mais defeitos do que leitura não dirigida
 - leitura baseada em cenários
 - como se comporta o sistema nas condições A, B, C, ... ?
 - leitura usando a visão do papel desempenhado pelo observador
 - ponto de vista do testador
 - como vou testar isso?
 - ponto de vista do mantenedor
 - é fácil manter isso?
 - as características (*features*) estão bem delimitadas?
 - ...
 - critérios explicitamente definidos a priori


Boehm, B.W.; Basili, V.R.; "Software Defect Reduction Top 10 List"; *IEEE Computer* 34(1); 2001; pags 135-137 – dizem que leitura dirigida captura 35% mais defeitos do que leitura normal

Mar 2015

Arndt von Staa © LES/DI/PUC-Rio

22

Pontos de vista, exemplos




Laboratório de Engenharia de Software

- Do ponto de vista do **testador**
 - para cada requisito discriminado na especificação: quais seriam os casos de teste que você utilizaria para testar o requisito?
 - está claro como determinar o resultado esperado?
- Do ponto de vista do **especificador**
 - existem potenciais conflitos entre os requisitos?
 - faltam requisitos, o serviço está completamente descrito?
 - além dos requisitos funcionais foram considerados os não funcionais?
- Do ponto de vista do **implementador**
 - quais seriam as estimativas de esforço para implementar cada requisito?
 - está claro o que é desejado?
 - os casos de uso têm dimensão moderada
 - requerem no máximo 3 dias para serem implementados

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
23

Pontos de vista, exemplos




Laboratório de Engenharia de Software

- Do ponto de vista do **usuário**
 - preciso realmente deste requisito?
 - falta algum requisito (funcional ou não) de que necessito?
 - entendi o que o requisito se propõe a fazer?
 - está de acordo com o que espero do serviço?
 - está de acordo com a legislação?
 - o tratamento de erros de uso faz sentido?
- Do ponto de vista da **prevenção de riscos**
 - a especificação, ou a arquitetura, ou o projeto, oferecem riscos (*design bad smells*) de levar a código contendo defeitos, deficiências ou vulnerabilidades?
 - o **sistema** pode causar elevados impactos negativos (prejuízos) caso venha a falhar ou ser utilizado de forma inesperada?

Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
24

Check list do ponto de vista do usuário




Laboratório de Engenharia de Software

- Marque todos os elementos cuja especificação não deixe clara a sua intenção
- Marque todos os elementos com especificação ambígua
 - diferentes leitores podem entender coisas diferentes
- Marque todos os elementos com especificação imprecisa
 - ex. “deve ter bom desempenho”
- Marque todas as redundâncias de especificação
 - diferentes elementos especificam a mesma coisa
- Marque todos os elementos cuja abrangência possa ser reduzida sem comprometer o propósito do artefato
- Marque todos os elementos que possam ser excluídos sem comprometer as necessidades e expectativas do serviço
- ...

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


Modos de revisar



Laboratório de Engenharia de Software

- Revisão pelo próprio autor** (*desktop checking*)
 - o autor lê e anota todos os problemas encontrados para, **depois**, removê-los
 - inconvenientes intrínsecos:
 - o autor tende a “ler o que acha que está escrito e não o que está de fato escrito”
 - erros de entendimento por parte do autor não são observáveis
 - o autor dirá que está correto segundo o que entendeu de forma incorreta
 - sujeito à síndrome da “ideia fixa” → insistir no erro
- O inconveniente pode ser atenuado utilizando **técnicas formais** (leves) para dirigir a leitura
 - a técnica formal induz uma **forma alternativa** (redundância útil) para observar o mesmo artefato


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

Modos de revisar


Laboratório de Engenharia de Software

- **Desenvolvimento em pares (duplas)**
 - *pair programming* proposto por *XP – eXtreme Programming*
 - o desenvolvimento de um artefato é realizado sempre por **duas pessoas** trabalhando em conjunto à frente de **um mesmo computador**
 - um digita, i.e. escreve o código
 - outro monitora o que está sendo escrito
 - observa erros de digitação
 - observa erros de uso dos elementos do programa
 - observa desvios com relação à especificação ou ao projeto
 - observa desatenção com relação a boas práticas, padrões e normas
 - propõe soluções alternativas melhores
 - tem sido utilizado com sucesso ao desenvolver código
 - trabalho em grupo é uma forma de desenvolvimento em “pares”?
 - em geral utilizado para especificar e arquitetar sistemas

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

Modos de revisar



Laboratório de Engenharia de Software

- **Revisão por pares (peer review)**
 - um dos parceiros (colega) do autor **lê e anota** todos os defeitos e demais problemas encontrados
 - ninguém deve ficar ofendido com as anotações, o que está em jogo é a qualidade do artefato!
 - ninguém deve deixar de anotar defeitos em virtude de algum receio de melindrar o autor
 - o que não impede de ser bem educado
 - defeitos e outros problemas devem ser anotados em um documento (formulário)
 - de preferência à parte → laudo explícito serve como controle do processo de garantia da qualidade
 - pequenas melhorias podem ser anotadas no próprio artefato
 - seria bom se existisse editor capaz de registrar alterações e anotações
 - » exemplo: registro de alterações em Word
- Revisão por pares: **par** aqui é no sentido de “colega”, “parceiro”, “pessoa de mesma condição ou nível social”

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

Laboratório de Engenharia de Software

Modos de revisar




- **Revisões progressivas** (*round-robin*)
 - seleciona-se um conjunto de parceiros que farão a revisão
 - cada um com uma determinada especialidade
 - cada um examinará a partir de um ponto de vista específico, exemplos de pontos de vista:
 - como se pode testar isso?
 - está de acordo com os padrões de projeto e os de programação?
 - está de acordo com os padrões das bases de dados da organização?
 - qual é o esforço para implementar isso?
 - o usuário precisa e entende isso?
 - ...
 - cada parceiro lê e anota todos os problemas encontrados dentro de sua especialidade
 - a seguir passa a diante para o próximo parceiro da lista

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


Laboratório de Engenharia de Software

Modos de revisar



- **Walk through** (caminhar através)
 - realizado em uma reunião envolvendo um ou mais colegas
 - o autor percorre o artefato sendo revisado e narra o comportamento esperado
 - os parceiros
 - indicam erros
 - expõem dúvidas
 - propõem melhorias
 - propõem soluções alternativas a serem avaliadas
 - propõem possíveis soluções quando o autor estiver em dúvida ou estiver enganado
 - o autor, ou um parceiro designado para ser o relator, registra os problemas e soluções propostas
 - **forma simplificada**: analista (psicólogo) do desenvolvedor ☺


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

Críticas às revisões


Laboratório de Engenharia de Software

- Prós
 - simplicidade
 - eficácia
 - apesar de informais, revisões tendem a encontrar um número significativo de defeitos
 - se feitas por pessoas treinadas em aspectos formais (argumentação da corretude), são muito mais eficazes
 - eficiência
 - em uma única revisão identifica-se uma quantidade grande de defeitos
 - baixo custo
 - o custo da revisão é amplamente compensado pela redução do retrabalho inútil
 - muitas coisas podem ser automatizadas
 - análise estática: *lint*, problemas potenciais, defeitos, inconsistências
 - compiladores hoje fazem uma boa parte do que se costumava fazer a mão

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

Críticas às revisões


Laboratório de Engenharia de Software


- Contras
 - a qualidade da revisão depende excessivamente da habilidade dos revisores
 - proficiência
 - cultura
 - a confiabilidade depende do rigor adotado pelo revisor
 - frequentemente não é repetível
 - revisor muda de opinião
 - diferentes revisores têm diferentes opiniões
 - a falta de treinamento dos revisores amplifica os problemas decorrentes da informalidade

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

Laboratório de Engenharia de Software

Inspeções

- Uma inspeção é uma **leitura crítica** do artefato e da documentação complementar
 - segundo um plano definido
 - obedecendo a regras de leitura definidas
 - envolvendo uma pequena equipe
- Inspeções diferenciam-se das revisões por serem mais formalizadas e serem mais eficazes
 - são também (bem?) mais caras




Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
33

Laboratório de Engenharia de Software

Inspeções, plano

- Plano de inspeção, etapas
 - **Produzir sinopse** (resumo)
 - autor redige uma descrição resumida do artefato a ser inspecionado
 - **Organizar a inspeção**
 - o autor seleciona a equipe de inspeção
 - o autor marca a data e horário da reunião de inspeção
 - **Realizar a leitura individual**
 - o autor distribui aos revisores o material a inspecionar e o de apoio
 - cada um dos revisores (pares) lê e anota os problemas observados
 - **Coletar e filtrar as observações em reunião de inspeção**
 - todos que concordaram devem comparecer e ser **pontuais**
 - resulta no **laudo da inspeção**
 - **Corrigir segundo o laudo**
 - o autor faz as correções segundo o laudo
 - **Acompanhar o progresso**
 - são registrados os eventos: distribuição, reunião, conclusão
 - é acompanhada a resolução dos problemas
 - se solicitada, é realizada nova inspeção após a correção



Mar 2015
Arndt von Staa © LES/DI/PUC-Rio
34

Inspeções, execução



- Durante a reunião
 - o autor narra o comportamento do artefato
 - os parceiros, que já devem ter lido a documentação fornecida
 - solicitam explicações
 - indicam a existência de problemas
 - conforme as anotações deles
 - os identificados durante a apresentação
 - sugerem melhorias
 - é produzido um laudo com todos os problemas e sugestões observados
- Após a reunião
 - autor faz as correções
 - possivelmente modifica coisas que não haviam sido anotadas
 - dependendo da gravidade, repete-se a inspeção

Inspeções, equipe



- Papéis desempenhados
 - **autor**
 - tem treinamento nos critérios a serem utilizados na avaliação
 - critérios visam aproximar-se do desenvolvimento correto por construção
 - desenvolve o artefato
 - idealmente segundo um processo definido e visando os critérios
 - escolhe a equipe de inspeção específica para o artefato e marca uma data para a reunião de inspeção
 - distribui com suficiente antecedência aos membros do comitê o material relativo ao artefato e as informações complementares
 - **inspetores**, dois ou três
 - têm treinamento nas técnicas e critérios utilizados na avaliação
 - recebem com antecedência o material a ser inspecionado
 - lêem e anotam
 - os defeitos encontrados
 - dúvidas e dificuldades de compreensão
 - possibilidades de melhorias

- Mar 2015

- Mar 2015

- Mar 2015

Mar 2015