


Laboratório de Engenharia de Software

Processos, Conceitos

Arndt von Staa
Departamento de Informática
PUC-Rio
Abril 2014

Especificação



Laboratório de Engenharia de Software

- Objetivos
 - Identificar características do processo de desenvolvimento a partir de um ponto de vista sistêmico.
- Justificativa
 - o processo de desenvolvimento é em última análise um sistema que transforma conhecimento e desejos (requisitos) em um sistema possuindo satisfatória qualidade de serviço e de engenharia.

Abr 2014

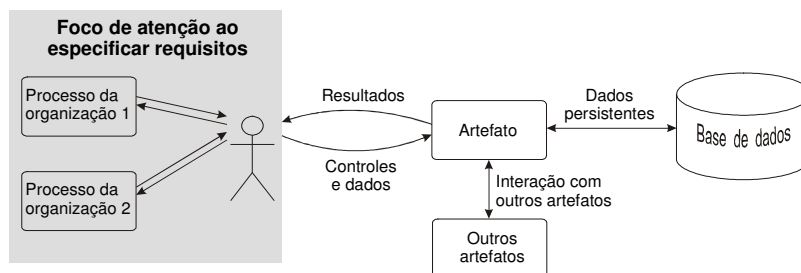
Arndt von Staa © LES/DI/PUC-Rio

2

Crença



- A percepção da qualidade de um software depende da adequação, eficiência, ... (fidedignidade) com que o software apoia o **processo em que o usuário** estiver atuando
 - isso põe em cheque a definição de qualidade “corresponde à especificação” subjacente aos modelos de maturidade
 - mas enfatiza a proposta ágil: “software que funcione”



Abr 2014

Arndt von Staa © LES/DI/PUC-Rio

3

Objetivo de um sistema



- Software deve agregar **valia** (*value*)
 - The **fundamental goal of all good design and engineering** is to create **maximal value added** for any given investment. There are many dimensions in which value can be assessed, from monetary profit to the solution of social problems.
 - Alguns aspectos de valia
 - benefícios econômicos, sociais, ecológicos, ...
 - necessidades atendidas
 - *time to market*
 - imagem da empresa
 - redução de riscos
 - lucro
 - SWOT (*strengths, weaknesses, opportunities, threats*)
 - ...

Valia: Utilidade, préstimo, serventia, importância, valor (Aurélio eletrônico)

Boehm, B.W.; Sullivan, K.J.; "Software Economics: a Roadmap"; Proceedings *ICSE'00 International Conference on Software Engineering*, Limerick, Ireland; New York: Association for Computing Machinery; 2000; pags 319-343

Abr 2014

Arndt von Staa © LES/DI/PUC-Rio

4

Realimentação contínua



- Metáfora
 - retroalimentação (*feedback*)
 - é razoável esperar que especificações sejam completas, corretas e imutáveis?
 - um desenvolvimento bem sucedido se dá por aproximações sucessivas
 - na realidade é uma forma de tentativa e erro
 - a diferença é que as tentativas não são aleatórias e, sim, são especificadas e avaliadas
 - especifica, faz, avalia, identifica correções / melhorias, volta a especificar
- Analogia
 - foguete que persegue um alvo móvel de rota incerta
 - ao invés de tiro de canhão disparado para atingir um alvo móvel de rota **supostamente** preditível

Abr 2014

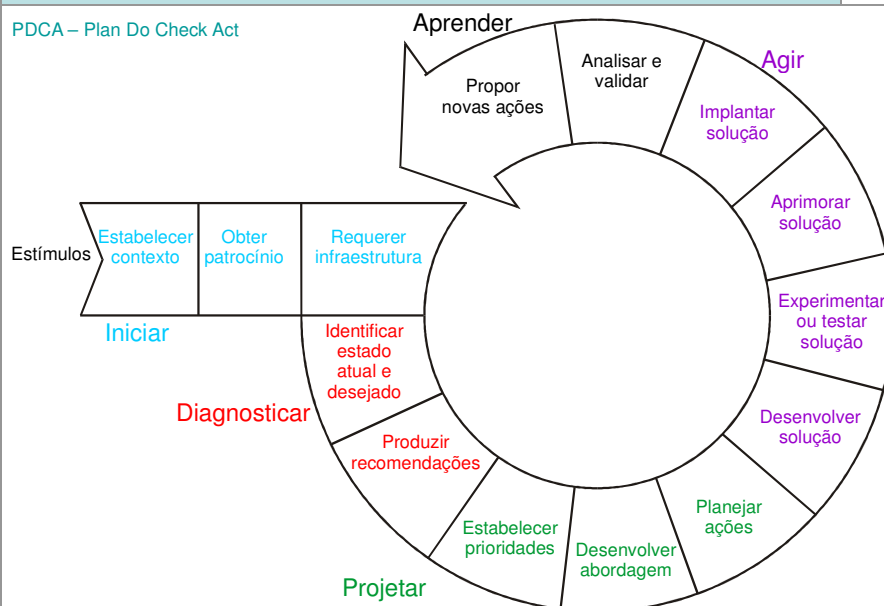
Arndt von Staa © LES/DI/PUC-Rio

5

Melhoria contínua: processo PDCA



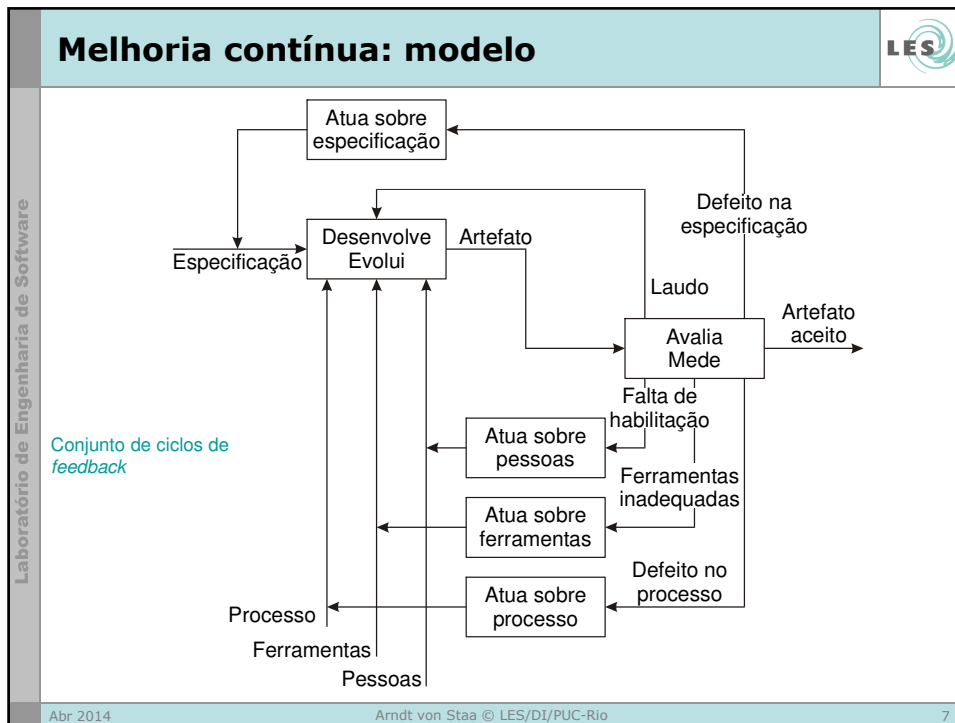
PDCA – Plan Do Check Act



Abr 2014

Arndt von Staa © LES/DI/PUC-Rio

6



Aprendizado contínuo

- Metáfora
 - Software é **conhecimento** adquirido gradativamente
 - Aprendemos a respeito do problema a resolver (domínio do problema; ou da aplicação)
 - Aprendemos a respeito da solução dada (domínio da tecnologia)
 - Desenvolvimento de software é um **processo de aprendizado**
 - o desenvolvimento termina quando se aprendeu o suficiente
 - Software é uma forma de **registro do conhecimento adquirido**
 - É também um **processo de aproximações sucessivas**
 - cada incremento constitui
 - » um pouco de aprendizado
 - » avaliação deste aprendizado
 - » correções de rota (alterações) em virtude do aprendizado

Abr 2014 Arndt von Staa © LES/DI/PUC-Rio 8

Aprendizado contínuo



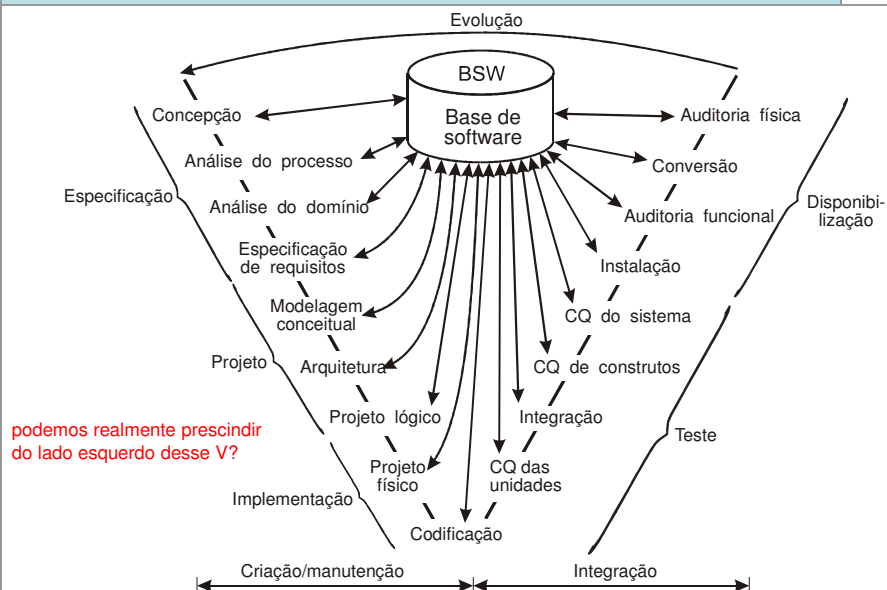
- Em geral o desenvolvimento prossegue do mais abstrato para o menos abstrato (mais concreto)
 - elaboração (**reengenharia**)
- Outras vezes parte do menos abstrato para o mais abstrato
 - reflexão (**engenharia reversa**)
- Na prática temos uma **negociação** entre o **desejável** (abstração) e o **realizável** (implementação)
- Finalmente, podem ocorrer **transformações** entre diferentes representações em um mesmo nível de abstração
 - consolidação visando assegurar consistência
 - validação da consistência

Abr 2014

Arndt von Staa © LES/DI/PUC-Rio

9

Etapas do processo de desenvolvimento

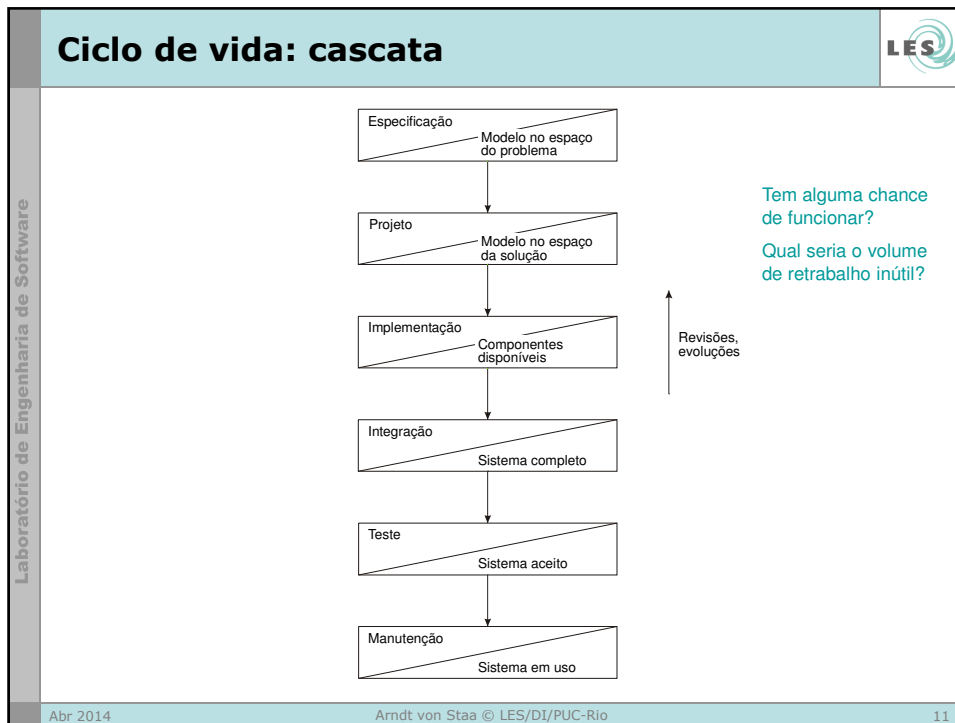


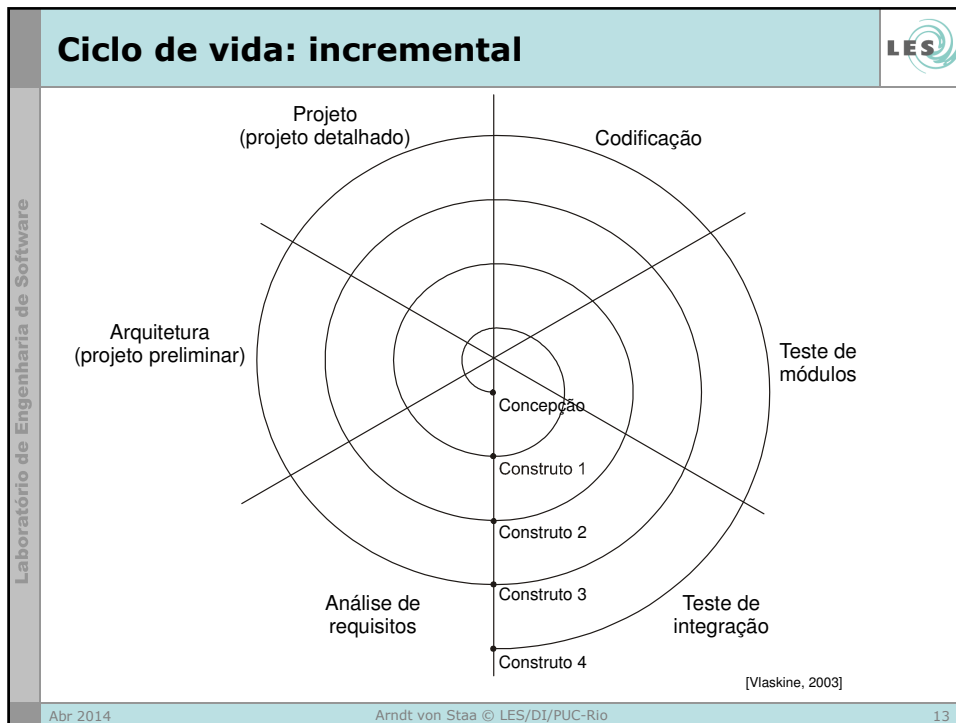
Staa, A.v.; Programação Modular; Rio de Janeiro: Campus; 2000

Abr 2014


Arndt von Staa © LES/DI/PUC-Rio

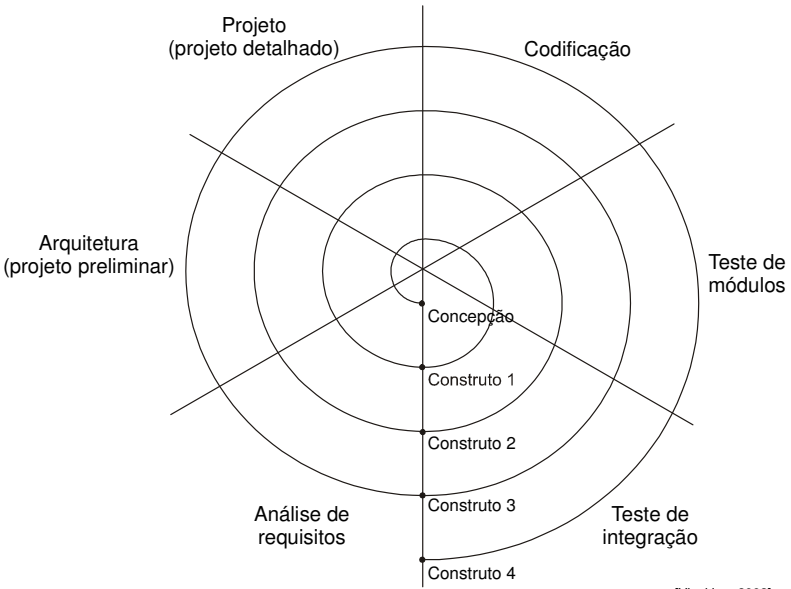
10





Desenvolvimento incremental





[Vlaskine, 2003]

- **Metáfora**
 - A grande caminhada: para andar mil quilômetros, dá-se um passo após o outro
 - Especifica (detalha, adiciona, muda) um pouco
 - Experimenta (prototipa, estuda) um pouco
 - Implementa um pouco
 - Aprimora e assegura a qualidade do incremento
 - Avalia a adequação, utilizabilidade, etc.
 - disponibiliza para o uso
 - Repete isso no próximo passo

Abr 2014 Arndt von Staa © LES/DI/PUC-Rio 14

Formas de incremento



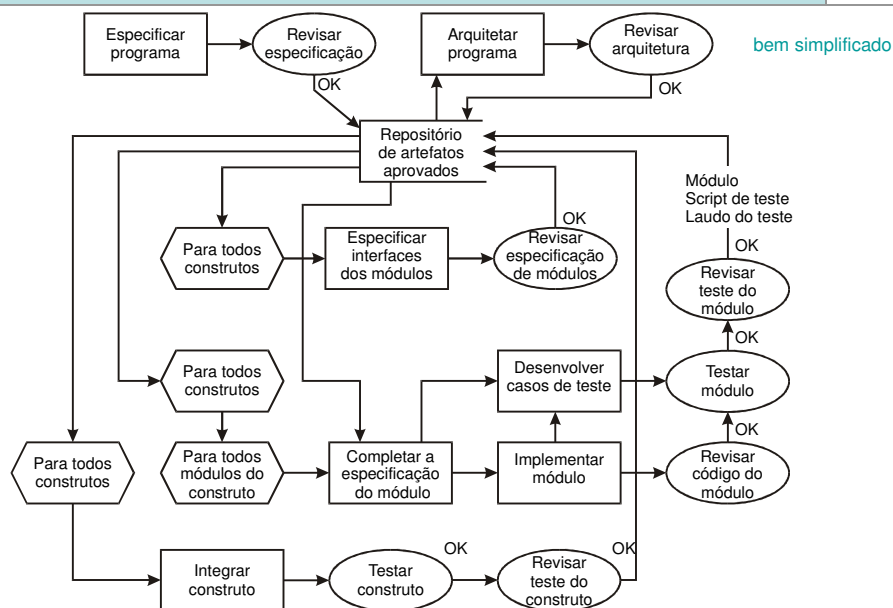
- Passos sucessivos do abstrato para o concreto
 - inspeções
 - protótipos gerados
 - criar testes, assertivas e outras formas de controle da qualidade junto como as especificações, arquiteturas e design
- Sucessivas implementações parciais
 - protótipos – soluções a serem descartadas, visam somente o entendimento do problema a resolver e de sua possível solução
 - **soluções reais** cada vez mais abrangentes
 - requer uma espécie de plano de liberações (*release plan*)

Abr 2014

Arndt von Staa © LES/DI/PUC-Rio

15

Desenvolvimento incremental, ciclos



Abr 2014

Arndt von Staa © LES/DI/PUC-Rio

16

Desenvolvimento incremental



- Para cada construto, componente e módulo
 - completa a especificação
 - propriedades funcionais e não funcionais
 - especifica como vai testar
 - desenvolve as suites de teste
 - desenvolve as assertivas
 - desenvolve controla a qualidade
 - integra
 - passos avançando na direção de componentes e/ou construtos
 - controla a qualidade do integrado parcial
 - termina formando um componente ou um construto
 - controla a qualidade do componente ou do construto
 - de engenharia: verificação, validação
 - de serviço (ponto de vista do usuário): aceitação


Desenvolvimento incremental



- Vantagens
 - menos retrabalho inútil
 - erros de especificação ou de design são descobertos cedo
 - adaptabilidade a mudanças de especificação e de design
 - visibilidade do progresso
 - existe sempre alguma coisa útil que poderá ser apresentada

Laboratório de Engenharia de Software

Desenvolvimento incremental



- Problemas com desenvolvimento incremental
 - liberações frequentes podem tornar necessário o retreinamento frequente dos usuários
 - usuário não vai gostar...
 - pode ocorrer a necessidade de mudança estrutural em bases de dados já povoadas
 - *refactoring*
 - de código – existe apoio de ferramentas
 - de design e de arquitetura – existem boas ferramentas?
 - cada incremento induz algum retrabalho em artefatos já concluídos
 - em geral útil – se devidamente planejado
 - sem ferramentas apropriadas – em particular teste automatizado – é virtualmente inviável


Abr 2014

Arndt von Staa © LES/DI/PUC-Rio

19

Laboratório de Engenharia de Software

Controle da qualidade




- Metáfora
 - **Errare humanum est**
 - Falibilidade é uma característica inata dos humanos,
 - consequentemente *não tenha medo de errar*
 - use **sempre** boas práticas
 - use **bons** métodos de verificação, validação e aceitação
 - » controle da qualidade passo a passo
 - ao encontrar um defeito elimine-o
 - » mesmo que tenha sido inserido por outro desenvolvedor
 - sendo possível use alguma forma de redundância
 - » redundâncias permitem a verificação automática do correto funcionamento
 - » assertivas pontuais, verificadores estruturais
 - » testes automatizados, criados antes do desenvolvimento

Abr 2014

Arndt von Staa © LES/DI/PUC-Rio

20

Controle da qualidade




Laboratório de Engenharia de Software

- Controle todos os aspectos da qualidade, algumas técnicas:
 - revisão ou inspeção sistemática, programação por pares
 - verificação de modelos
 - análise estática
 - medição de propriedades estáticas
 - teste, idealmente automatizado
 - medição de propriedades dinâmicas
 - instrumentação
 - módulos dublê

Abr 2014
Arndt von Staa © LES/DI/PUC-Rio
21

Evite a injeção de defeitos: formalismo




Laboratório de Engenharia de Software

- Uso de **técnicas formais leves**
 - assertivas, sem preocupação com sua automação
 - *design by contract*
 - argumentação da correteza
 - verificação de modelos
 - medição de modelos
 - análise estática de código
 - medição estática de código
- Software **auto-verificante** (*self-checking*)
 - adição de redundância
 - assertivas executáveis
 - verificação pontual
 - verificação estrutural

Abr 2014
Arndt von Staa © LES/DI/PUC-Rio
22

Laboratório de Engenharia de Software

Controle dinâmico da qualidade



- Desenvolvimento de técnicas de garantia da corretude do programa em execução
- Software orientado à recuperação
 - auto-verificação (*self-checking*)
 - adição de redundância
 - assertivas executáveis
 - verificação pontual
 - verificação estrutural
 - medição do comportamento dinâmico
 - self-recovery
 - self-healing
- Aplicação dessas idéias em software real


Abr 2014

Arndt von Staa © LES/DI/PUC-Rio

23

Laboratório de Engenharia de Software

Experimentação e medição



- Estudo de métricas e de processos de medição
- Condução de experimentos e medições no contexto de orientação a aspectos


Abr 2014

Arndt von Staa © LES/DI/PUC-Rio

24

Laboratório de Engenharia de Software

Evolução da qualidade




- Metáfora:
 - Entropia crescente
 - De maneira geral alterações sucessivas levam à desestruturação do conjunto de representações
 - degeneração arquitetural
 - » Hochstein, L.; Lindvall, M.; "Diagnosing Architectural Degeneration"; *28th Annual NASA Goddard Software Engineering Workshop*, Greenbelt, MD; Los Alamitos, California: IEEE Computer Society; 2003; pags 137-142
 - » Tvedt, R.T.; Costa, P.; Lindvall, M.; "Does the Code Match the Design? A Process for Architecture Evaluation"; *18th IEEE International Conference on Software Maintenance (ICSM'02)*; Los Alamitos, CA: IEEE Computer Society; 2002; pags 393-401
 - decaimento do código (*code decay*)
 - » Eick, S.G.; Graves, T.L.; Karr, A.K.; Marron, J.S.; Mockus, A.; "Does Code Decay? Assessing the Evidence from Change Management Data"; *IEEE Transactions on Software Engineering* 27(1); Los Alamitos, California: IEEE Computer Society; 2001; pags 1-12

Abr 2014
Arndt von Staa © LES/DI/PUC-Rio
25

Laboratório de Engenharia de Software

Evolução da qualidade




- Sempre reveja a organização (refactoring) 1/2
 - de cada artefato
 - representação
 - do conjunto de representações
- Corrija-a sempre que necessário
 - Lazy evaluation – sempre que for realizar alguma manutenção
 - há os que recomendam corrigir sempre que se observe alguma deficiência estrutural
 - custo disso compensa?
 - Refactoring de código
 - Refactoring de arquitetura e do design

Abr 2014
Arndt von Staa © LES/DI/PUC-Rio
26

Laboratório de Engenharia de Software

Evolução da qualidade




- **Sempre reveja a organização** (refactoring) 2/2
 - deixar de rever e corrigir **tende** a criar problemas mais adiante
 - à medida que o sistema vai sendo mantido mais e mais módulos participam de cada passo de manutenção
 - esforço de manutenção crescente
 - para um mesmo volume e complexidade de funcionalidade alterada
- Evidentemente artefatos de expectativa de vida curta são menos vulneráveis à degradação decorrente da manutenção
 - e se forem reusados?

Abr 2014
Arndt von Staa © LES/DI/PUC-Rio
27

Laboratório de Engenharia de Software

Consequência: alvo móvel




- De maneira geral software **converge** para a solução desejada → software amadurece
 - a velocidade da convergência depende do conhecimento prévio adquirido
 - experiência profissional
 - *frameworks*
 - componentes
 - protótipos
 - a convergência se dá tanto ao **desenvolver** como ao **manter**
 - maturação do software
 - o decaimento e a degeneração impedem a convergência, podendo chegar a impossibilitar a evolução
 - software precisa ser manutenível (evolutível), verificável e validável (testável)

Abr 2014
Arndt von Staa © LES/DI/PUC-Rio
28

Laboratório de Engenharia de Software

Consequência: alvo móvel

- A solução desejada evolui
 - o próprio software muda o contexto em que está inserido, levando ao desejo de evolução da solução, uma vez que o novo contexto conduz a requisitos novos ou modificados.
- Retornando ao início: o desenvolvimento e o uso pode ser entendido como um **processo de aprendizado**




Ramil, J.F.; Lehman, M.M.; "Software Evolution and Software Evolution Process"; *Annals of Software Engineering* 14; Dodrecht: Kluwer; 2002; pags 275-309

Abr 2014
Arndt von Staa © LES/DI/PUC-Rio
29

Laboratório de Engenharia de Software

Bibliografia

- Beck, K.; "The Inevitability of Evolution"; *IEEE Software* 27(4); 2010; pags 28-29
- Boehm, B.W.; "The Changing Nature of Software Evolution"; *IEEE Software* 27(4); 2010; pags 26-28
- Boehm, B.W.; Sullivan, K.J.; "Software Economics: a Roadmap"; Proceedings *ICSE'00 International Conference on Software Engineering*, Limerick, Ireland; New York: Association for Computing Machinery; 2000; pags 319-343
- Ramil, J.F.; Lehman, M.M.; "Software Evolution and Software Evolution Process"; *Annals of Software Engineering* 14; Dodrecht: Kluwer; 2002; pags 275-309
- Staa, A.v.; *Programação Modular*; Rio de Janeiro: Campus; 2000



Arndt von Staa © LES/DI/PUC-Rio

Abr 2014
Arndt von Staa © LES/DI/PUC-Rio
30