


Laboratório de Engenharia de Software

Teste baseado em riscos

Arndt von Staa
Departamento de Informática
PUC-Rio
Março 2014

Especificação



Laboratório de Engenharia de Software

- Objetivo desta aula
 - discutir algumas técnicas para a redução do custo do teste, sem impactar *significativamente* a fidedignidade
- Justificativa
 - Testes devem enfatizar os casos de teste relevantes
 - procurar eliminar os defeitos e vulnerabilidades de **maior risco**
 - ex. código inseguro
 - Caso os critérios de geração de massas de teste sejam levados ao pé da letra será frequentemente gerado um número muito grande casos de teste. Consequências:
 - custo elevado
 - comprometimento do tempo disponível para os testes
 - possivelmente vários casos pouco relevantes

Mar 2014Arndt von Staa © LES/DI/PUC-Rio2

Objetivo do desenvolvimento



- **Objetivo do desenvolvimento:** atingir **fidedignidade satisfatória** de forma **econômica**
 - a fidedignidade está relacionada com a capacidade de
 - **injetar poucos** defeitos **relevantes** ao desenvolver e manter
 - **detectar e eliminar** a (quase-) totalidade dos *defeitos relevantes* antes de por ou repor em uso
 - manter fidedignidade satisfatória **durante toda a vida** útil do software
 - e conseguir isso tudo com
 - alta **produtividade**
 - dimensão / esforço, ex. funcionalidades entregues por homem.hora
 - **custo compatível** com a *valia* (*value*) do artefato
 - nem sempre baixo custo é o desejável
 - » procure assegurar baixo custo total
 - o custo é fortemente afetado pelo **retrabalho inútil**, que, por sua vez, é afetado pela **densidade de defeitos inicial** ao desenvolver ou manter
- valia* - 3.Utilidade, préstimo, serventia, valência, valimento, valor; [Aurélio eletrônico]

Mar 2014

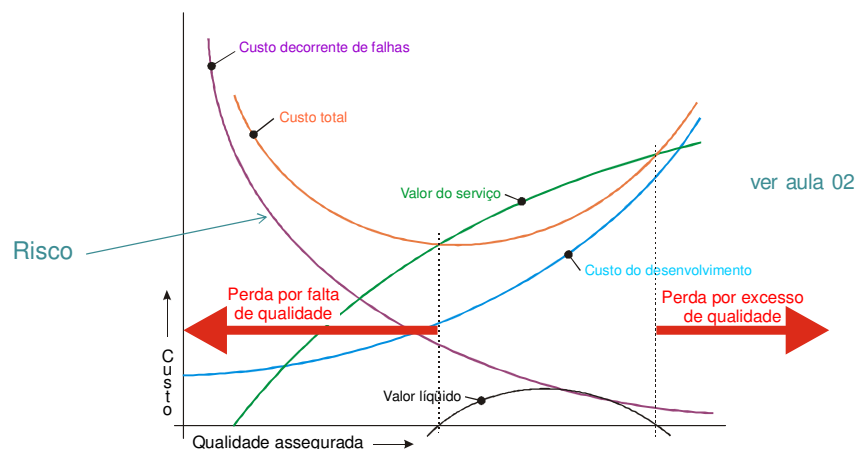
Arndt von Staa © LES/DI/PUC-Rio

3

Objetivo dos testes, recordação



- O objetivo do teste é **encontrar** e **promover a remoção** das causas de **falhas relevantes**, dessa forma evitar **lesões**
- Falhas relevantes possuem **riscos elevados**



Mar 2014

Arndt von Staa © LES/DI/PUC-Rio

4

Laboratório de Engenharia de Software

Objetivo dos testes, recordação

- O que entendemos por “encontrar e promover a remoção”?
 - O teste encontra falhas e vulnerabilidades e produz um **laudo**
 - teste não corrige, nem melhora qualidade
 - o laudo contém os **relatos das falhas** e das **vulnerabilidades** encontradas
 - vulnerabilidades decorrem da existência de algum defeito
 - O relato de falha deve conter informação que permita **diagnosticar** a causa da falha ou vulnerabilidade
 - O relato de falha deve incentivar a **remoção do defeito** causador
 - deve indicar a prioridade de remoção
 - Somente a **remoção completa e correta** dos defeitos conhecidos é que melhora a qualidade

LES

Mar 2014
Arndt von Staa © LES/DI/PUC-Rio
5

Laboratório de Engenharia de Software

Observação prática

- **Defeitos problemáticos provocam repetidamente falhas**
 - tais defeitos tendem a ser removidos com o passar do tempo
 - *amadurecimento* leva à eliminação de defeitos
 - mas *manutenção* (evolução) adiciona novos defeitos
- **Muitos defeitos remanescentes têm chance virtualmente 0 de serem exercitados**
 - poucos defeitos remanescentes possuem risco alto
 - estudo IBM dos **defeitos remanescentes** somente 2% provocaram falhas relevantes recorrentes

Total de defeitos remanescentes

Defeitos que oferecem riscos

Hatton, L.; “Exploring the Role of Diagnosis in Software Failure”; IEEE Software 18(4); Los Alamitos, CA: IEEE Computer Society; 2001; pags 34-39

LES

Mar 2014
Arndt von Staa © LES/DI/PUC-Rio
6

Controle do custo do teste



- Procure **injetar** poucos defeitos ao desenvolver
 - elevada proficiência da equipe
 - contribui para uma significativa redução da injeção de defeitos nos variados artefatos gerados ao desenvolver ou manter
 - revisões ou inspeções realizadas com regularidade
 - contribuem para a identificação e eliminação de defeitos cedo no desenvolvimento ou manutenção
 - desenvolvimento e integração incremental
 - contribuem para a redução de problemas nas especificações, na arquitetura e nos projetos
 - boas práticas
 - contribuem, por construção, para a redução da injeção de defeitos
 - boas ferramentas
 - contribuem para evitar ou para observar defeitos
 - padrões eficazes
 - eliminam ou reduzem, por construção, a frequência de determinadas classes de defeitos

Controle do custo do teste



- Procure reduzir o **custo do teste**
 - concentrar nos casos de teste relevantes
 - reduzir o número de casos de teste pouco relevantes
 - aumentar a eficácia dos testes
 - prover capacidade de detectar falhas e diagnosticar os defeitos causadores
 - aumentar a eficiência dos testes
 - reduzir o número de vezes que testes são bem sucedidos ao serem **reexecutados**
 - um **teste bem sucedido** é um que encontra alguma falha
 - automatizar o que for possível
 - automação da execução dos casos de teste
 - elimina a necessidade de testadores humanos
 - automação da geração dos casos de teste úteis
 - reduz significativamente o esforço ao elaborar casos de teste úteis

Sequência de testes eficazes



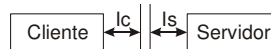
- Testar unidades **com rigor**
 - desenvolver testes antes de ou junto com as unidades
 - utilizar critérios de teste estruturais
 - assegurar elevada cobertura do código
 - teste de unidade é possivelmente a única oportunidade de alcançar elevados graus de cobertura envolvendo o código

Sequência de testes eficazes

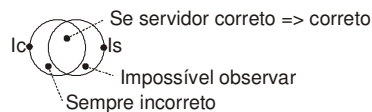


- Testar componentes
 - desenvolver testes dos componentes antes de ou junto com a integração dos componentes
 - testar as condições das interfaces dos componentes
 - assegurar elevada cobertura de chamadas e retornos
 - é possivelmente a única oportunidade de alcançar elevados graus de cobertura de cenários envolvendo as condições das interfaces conceituais
 - interface conceitual: todos parâmetros, dados retornados, variáveis membro de objeto, variáveis globais, variáveis persistentes

– problema:




Ic - interface do ponto de vista do cliente
Is - interface do ponto de vista do servidor



Laboratório de Engenharia de Software

Sequência de testes eficazes




- Testar características (*features*) elementares
 - desenvolver testes funcionais antes de ou junto com a disponibilização de características
 - testar característica a característica à medida que forem disponibilizadas
 - assegurar elevada cobertura da funcionalidade e da interface humana de cada característica
 - oportunidade para testar uma variedade de cenários de uso, inclusive cenários não convencionais ou não esperados, e.g. erros de uso
 - medir progresso do projeto observando a conclusão de características
- Isto requer um plano de liberações → *release plan*
 - a sequência de liberações que leva ao sistema completo
 - as características que compõem cada liberação

Mar 2014
Arndt von Staa © LES/DI/PUC-Rio
11

Laboratório de Engenharia de Software

Sequência de testes eficazes




- Testar a integração das características em um sistema
 - especificar e desenvolver testes de aceitação do sistema
 - realizar testes de usabilidade
 - realizar testes dos requisitos não funcionais
 - deveriam ser abordados em todas as etapas
 - assegurar elevada cobertura das necessidades do usuário, tanto no que tange o contexto de uso, como as necessidades não funcionais

Mar 2014
Arndt von Staa © LES/DI/PUC-Rio
12

Laboratório de Engenharia de Software

Volume dos testes



- Como manter o volume das massas de teste sob controle?
 - suítes de teste podem levar à **explosão combinatória**
- **Explosão combinatória** é o crescimento multiplicativo, ou exponencial, em função do número de **decisões, repetições e características interdependentes**
 - Exemplo: uma tabela de decisões com n condições pode levar a 2^n casos de teste distintos.


Mar 2014

Arndt von Staa © LES/DI/PUC-Rio

13

Laboratório de Engenharia de Software

Volume dos testes



- Precisamos de uma técnica para redução do volume de testes sem que isso reduza a fidedignidade a ponto de comprometer a garantia da qualidade
- Que tal enfatizar testes que têm por objetivo levar à remoção de defeitos relevantes caso existam?

Mar 2014


Arndt von Staa © LES/DI/PUC-Rio

14

Laboratório de Engenharia de Software

Defeito relevante

- **Defeito relevante** é um que causa uma falha relevante
 - leva a **alto risco** de uso
 - danos materiais, prejuízo financeiro
 - danos pessoais
 - ...
 - leva a **alto risco** de desenvolvimento ou manutenção
 - custo ao desenvolver realizado muito maior do que o estimado
 - prazo para desenvolver realizado muito maior do que o estimado
 - projeto cancelado
 - resultado do desenvolvimento descartado
 - resultado do desenvolvimento "buguento"
 - ...
- Um **caso de teste relevante** é capaz de observar falhas relevantes e promover a remoção completa e correta dos defeitos causadores




Mar 2014
Arndt von Staa © LES/DI/PUC-Rio
15

Laboratório de Engenharia de Software

Por que se preocupar com risco?

- A **análise de riscos** relacionado aos erros leva em conta:
 - a **probabilidade da ocorrência** de um erro
 - será tão maior quanto
 - mais **relaxado** for o controle da qualidade
 - menos se procurar desenvolver **software correto por construção**
 - menor for a **proficiência** dos desenvolvedores
 - menor for o conhecimento relativo ao **domínio da aplicação**
 - mais **depende da correteude** das bibliotecas e da plataforma
 - mais depender do **correto uso pelo usuário**
 - o **impacto do erro**
 - a dimensão do dano causado pela falha
 - note que erro não observado pode gerar grandes lesões
 - precisamos assegurar que erros serão observados, reportando falhas
 - a **relevância** do serviço ou artefato afetado pela falha
 - a importância que serviço ou artefato têm para a organização ou para o usuário
 - exemplos: ferramentas, bibliotecas, arcabouços, serviços essenciais



Mar 2014
Arndt von Staa © LES/DI/PUC-Rio
16

Por que se preocupar com risco?



- Quanto maior for o **dano** ou a **relevância**
 - muito menor deve ser a **probabilidade** de defeitos ou vulnerabilidades persistirem no sistema
 - logo: muito menor deve ser a **probabilidade** de passar despercebido o erro consequente de um defeito
- Mas as causas dos erros – i.e. os defeitos – são
 - desconhecidas
 - se fossem conhecidas, poderiam ter sido removidas, óbvio
 - intrinsecamente inevitáveis
 - o desconhecimento e/ou a falibilidade dos desenvolvedores injeta defeitos
 - especificações defeituosas levam a sistemas contendo defeitos
 - causas exógenas exploram, intencionalmente ou não, as vulnerabilidades (defeitos) existentes
 - usuários são humanos e, portanto, podem usar de forma incorreta

Risco, definição



- Componentes da especificação de um risco
 - nome do evento causador → a **ameaça**
 - extravasão de campo
 - não possui escalabilidade
 - ...
 - **impacto**
 - dano direto
 - dano indireto, i.e. dano propagado para outros artefatos ou sistemas
 - ...
 - **probabilidade**, exemplos que aumentam a probabilidade
 - especificação ruim
 - engenharia ruim, ex. arquitetura, projeto ruins ou até ausentes
 - alta densidade de defeitos e vulnerabilidades observada ao testar
 - teste mal feito
 - ...
 - **relevância**, exemplos que aumentam a relevância
 - artefato é central para o negócio
 - artefato é usado com frequência
 - artefato é componente de ou interage com vários sistemas
 - ...

Laboratório de Engenharia de Software

Tratamento do risco que se materializou

LES

- Para tratar eventos de **risco de uso** que tenham ocorrido é necessário
 - ser capaz de observar a falha
 - os eventos de risco de uso se manifestam através de algum erro, ou seja comportamento ou estado diferente do que deveria ser segundo a especificação (ou expectativas do usuário)
 - ser capaz de diagnosticar a falha
 - defeito contido no artefato
 - erro de uso não controlado
 - agressão possível devido a alguma vulnerabilidade
 - ser capaz de eliminar completamente a causa, ou ser capaz de controlar adequadamente a causa
 - quando não é possível eliminar a causa, torna-se necessário reduzir a significativamente probabilidade de ocorrência do evento
 - ser capaz de por a versão corrigida em operação

Mar 2014

Arndt von Staa © LES/DI/PUC-Rio

19

Laboratório de Engenharia de Software

Teste baseado em riscos

LES

(RBT- risk based testing)

Processo básico

```

graph LR
    A[Identificar riscos] --> B[Avaliar riscos]
    B --> C[Mitigar riscos]
    C --> D[Relatar riscos]
    D --> E[Predizer riscos]
    E -.-> A
    D -.-> F[Planejar tratamento]
    F -.-> C
    
```

Adaptado de: Karolak, D.W.; *Software Engineering Risk Management*; New York: Wiley-IEEE Computer Society Press; 1995; apud Amland 1999

Mar 2014


Arndt von Staa © LES/DI/PUC-Rio

20

Laboratório de Engenharia de Software

Identificar riscos

- Riscos são as consequências da **ocorrência** de eventos
 - o nome do evento é o nome do risco
- Riscos são em geral **indesejáveis**, exemplos
 - execução de um defeito
 - precisa-se verificar se existe um defeito da classe correspondente ao risco e então eliminá-lo
 - exploração bem sucedida de uma vulnerabilidade, ver: defeito
 - consequência de uma ação incorreta do usuário ou de algum serviço com que o sistema em questão interage
 - precisa-se verificar se existe a vulnerabilidade e então eliminá-la
- Observação
 - podem existir riscos desejáveis
 - exemplo: o custo do desenvolvimento ser significativamente menor do que o orçado




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

Laboratório de Engenharia de Software

Identificar riscos


- Criar um catálogo genérico dos potenciais riscos
 - este catálogo é utilizado para dirigir a identificação dos riscos a serem controlados no software em questão
 - é possível extravasar campos?
 - é possível injetar comandos SQL ao fornecer dados em um browser?
 - arquivos temporários permanecem disponíveis?
 - o conteúdo de arquivos excluídos continua disponível?
 - é possível acessar arquivos de outras aplicações?
 - é possível acessar arquivos na versão errada?
 - dados confidenciais podem ser acessados por não autorizados?
 - exemplo não computacional: o lixo gerado contém dados confidenciais
 - é possível vender mais de uma vez um mesmo produto?
 - é possível informar falta de estoque quando o item ainda existe em estoque?
 - ...



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

Laboratório de Engenharia de Software

Identificar riscos



- Selecionar o **catálogo do sistema** baseado nos casos de uso
 - sempre do ponto de vista da empresa, perguntas do gênero:
 - para cada dado a ser fornecido pelo usuário
 - o que ocorre se o usuário fornece um dado errado
 - o que vem a ser um dado errado?
 - o que vem a ser um dado **não plausível**?
 - usuário tenta fraudar
 - usuário tenta invadir
 - mais de N usuários tentam usar simultaneamente
 - para cada link
 - link para URL não existente
 - para cada ação
 - atividade realiza um cálculo errado
 - atividade cancela o processamento
 - usuário interrompe a transação antes de concluir
 - processamento é interrompido antes de concluir
 - » falta de energia, quebra ou falha de equipamento, logout
 - ...


Mar 2014

Arndt von Staa © LES/DI/PUC-Rio

23

Laboratório de Engenharia de Software

Identificar riscos



- Selecionar o **catálogo do sistema** baseado em critérios de qualidade
 - requisitos de disponibilidade
 - requisitos de desempenho
 - requisitos de escalabilidade
 - requisitos de capacidade
 - requisitos de manutenibilidade
 - requisitos de localizabilidade
 - requisitos de qualidade de engenharia
 - o que vem a ser qualidade de engenharia?
 - ...

Localizar um software: traduzir todas as mensagens, diálogos, menus, para um novo idioma de determinada sociedade (país), e adaptar o software à cultura correspondente


Mar 2014

Arndt von Staa © LES/DI/PUC-Rio

24

Laboratório de Engenharia de Software

Avaliar riscos




- Identificar os riscos relevantes do software em questão
 - reunião com cliente e usuários
- Criar uma planilha com as colunas:
 1. Nome da vulnerabilidade que caracteriza o risco
 2. Probabilidade da ocorrência de defeitos no artefato em questão
 - muito baixo 1, baixo 2, normal 3, alto 4, muito alto 5
 3. Impacto no uso do artefato em questão
 - muito baixo 1, baixo 2, normal 3, alto 4, muito alto 5
 4. Relevância do artefato em questão
 - muito baixo 1, baixo 2, normal 3, alto 4, muito alto 5
- Calcular o valor agregado do risco $VA = P * I * R$

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

Laboratório de Engenharia de Software

Avaliar riscos



- Modelo **pseudo** matemático ☺ do valor agregado

$$VA = \text{probabilidade} * \text{impacto} * \text{relevância}$$
- Probabilidade ::
 - {5 – muito alta , 4 – alta , 3 – aceitável , 2 – baixa , 1 – muito baixa}
- Impacto
 - {5 – muito alto , 4 – alto , 3 – aceitável , 2 – baixo , 1 – muito baixo}
- Relevância
 - {5 – muito alta , 4 – alta , 3 – aceitável , 2 – baixa , 1 – muito baixa}
- Nível do risco
 - {125 – 101 desastroso , 100 – 76 altíssimo , 75 – 51 muito alto , 50 – 26 alto , 25 – 10 aceitável , 9 – 1 irrelevante }


Chutologia? Qual seria um modelo melhor?

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

Laboratório de Engenharia de Software

Avaliar riscos

- Ordenar a planilha em ordem decrescente de VA
- Determinar os pontos de corte
 - Evitar sempre $VA \geq 75$,
 - Evitar se possível $25 < VA < 75$
 - Ignorar $VA \leq 25$
- Desenvolver testes cuidadosos
 - para os riscos a “evitar sempre”
 - se existirem recursos
 - para os riscos a “evitar se possível”
- É conveniente a planilha existir antes de se iniciar o desenvolvimento
 - reduz o esforço para controlar coisas de baixo risco
 - o **risco influencia a arquitetura e o projeto**




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

Laboratório de Engenharia de Software

Planejar tratamento

- Desenvolver planos de contingência
 - o que fazer se o evento identificado pelo risco ocorrer?
 - quando em teste, ex.
 - gerar uma solicitação de correção emergencial
 - ou gerar uma solicitação para uma futura versão
 - quando em uso
 - procedimentos de registro e tratamento de incidentes (falhas) observados
- Desenvolver planos para **mitigar**
 - como proceder para manter o dano sob controle no caso do evento ocorrer?
 - quando em teste
 - quando em uso



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

Mitigar riscos



- Mitigar o risco tem por objetivo
 - prevenir (impedir) a ocorrência do evento que oferece risco
 - ou, se isto não for possível, manter sob controle o dano consequente da ocorrência do evento
- Para poder mitigar é necessário ser capaz de observar a ocorrência do evento
 - caso a mitigação seja feita corretamente, não ocorrerão lesões
 - recordação: lesão é a ocorrência de um dano não conhecido
 - mas poderão ocorrer danos mensuráveis

Mitigar: abrandar, suavizar, diminuir (o impacto) [Aurélio eletrônico]

Mitigar riscos



- Para poder mitigar é necessário ser capaz de observar erros
 - como observar que o evento ocorreu ou está prestes a ocorrer?
 - como proceder para evitar a ocorrência do evento?
 - procurar tornar inexistentes os defeitos associados ao risco
 - como proceder para mostrar que o evento efetivamente tem baixa probabilidade de ocorrer?
 - testar, inspecionar
 - para fins de teste, como proceder para simular ou provocar a ocorrência do evento?
 - caso ocorra o evento, proceder como planejado
 - os grandes desastres tendem a ser a consequência da composição de vários eventos e de erros humanos ao tratar alguns deles
 - muitas vezes espera-se de humanos comportamento não humano, por exemplo, humanos podem errar, erram mais quando sob stress
 - ou seja, erro humano pode ser induzido por sistema com usabilidade inadequada

Relatar riscos



- Relatam-se os
 - eventos que ocorreram
 - podem-se observar novos riscos
 - a frequência com que ocorreram
 - as consequências da ocorrência
 - impactos (danos) observados
- Durante os testes devem ser medidos:
 - número de defeitos e vulnerabilidades encontrados
 - número de defeitos por funcionalidade
 - número de ocorrências de cada evento
 - tempo (horas, ou fração) gastas para encontrar a falha
 - tempo (horas, ou fração) gastas para eliminar o defeito
 - classificação do defeito
 - o evento a que corresponde


Predizer riscos



- Baseado nas medições realizadas (passado) prediz-se a possibilidade da ocorrência dos eventos associados a riscos
 - como consequência da predição pode-se concluir que determinadas funcionalidades (ou componentes) devem ser revistas
 - a mesma coisa aplica-se à arquitetura e aos projetos

Laboratório de Engenharia de Software

Manutenção do catálogo



- Um catálogo incompleto leva à perda de confiança
 - durante as diversas etapas do processo podem ser identificados novos riscos
 - devem ser incorporados ao catálogo
 - ao ler literatura sobre riscos, novos riscos podem ser identificados
 - devem ser incorporados ao catálogo
- Um catálogo muito extenso torna-se um estorvo
 - de tempos em tempos o catálogo deve ser revisto
 - riscos não mais observáveis devem ser transferidos para uma região de riscos “obsoletos” (deprecados)
 - riscos similares devem ser fundidos em um único
 - descrições de riscos devem ser revistas para assegurar atualidade e coerência com a terminologia atual

Mar 2014
Arndt von Staa © LES/DI/PUC-Rio
33

Laboratório de Engenharia de Software

Bibliografia



- A presente aula foi fortemente baseada nos textos a seguir
 - Amland, S.; *Risk Based Testing and Metrics*; 5th International Conference EuroSTAR '99; 1999, Barcelona, Spain
 - Bach, J.; *Heuristic Risk-Based Testing*; Software Testing and Quality Engineering Magazine, 11/99
 - Schaefer, H.; *Risk based testing, how to choose what to test more and less*; Notas de palestra; Software Test Consulting, Norway; 2004
 - Teunissen, R.; *Risk Based Test Strategy*; Notas de palestra; São Paulo; 2010; Polteq IT Services BV
- James R. Persse, J.R.; A Basic Approach to ITIL Service Operation; Atlanta: Tree Of Press; 2010; Kindle Edition

Mar 2014
Arndt von Staa © LES/DI/PUC-Rio
34

Laboratório de Engenharia de Software

LES

Fim

Mar 2014

Arndt von Staa © LES/DI/PUC-Rio

35