




Laboratório de Engenharia de Software

Teste Funcional 1

Arndt von Staa
Departamento da Informática
PUC-Rio
Março 2017


Especificação




Laboratório de Engenharia de Software

- Objetivo da aula
 - descrever o que são critérios de teste funcional e apresentar técnicas simples de apoio a essa categoria de testes
- Justificativa
 - testes funcionais são testes caixa fechada e são utilizados para verificar a existência de inadequações ao uso
 - são criados a partir das especificações
 - devem levar em conta o ponto de vista do usuário
 - as expectativas do usuário podem ser (usualmente são) diferentes das especificações
- Texto
 - Pezzè, M.; Young, M.; *Teste e Análise de Software*; Porto Alegre, RS: Bookman; 2008; capítulo 10; seção 14.3

Mar 2017Arndt von Staa © LES/DI/PUC-Rio2

Laboratório de Engenharia de Software	Especificação recordação		
	<ul style="list-style-type: none"> • A especificação deve estabelecer o que um artefato deve fazer do ponto da vista dos diversos interessados (stakeholders) <ul style="list-style-type: none"> – requisitos funcionais estão relacionados com as tarefas do usuário apoiadas pelo artefato <ul style="list-style-type: none"> • o usuário faz parte do sistema intensivo em software – requisitos não funcionais estão relacionados com o modo de resolver, ou com as propriedades da solução <ul style="list-style-type: none"> • muitas vezes chamados de requisitos de qualidade (da solução) – requisitos inversos coisas que o sistema não deve fazer <ul style="list-style-type: none"> • riscos que não podem ser assumidos – requisitos de interface humana <ul style="list-style-type: none"> • e também requisitos de interface com equipamentos e sistemas ou componentes externos 		
	Mar 2017	Arndt von Staa © LES/DI/PUC-Rio	3

Laboratório de Engenharia de Software	O que é teste funcional?		
	<ul style="list-style-type: none"> • O teste funcional tem por objetivo descobrir discrepâncias entre o que o artefato faz e a sua especificação funcional <ul style="list-style-type: none"> – o foco é testar a satisfação dos requisitos funcionais – é um teste caixa fechada (caixa preta) – assume especificações <i>corretas</i> e <i>completas</i> • Uma forma ampliada do teste funcional é o teste de aceitação que tem por objetivo descobrir discrepâncias entre o que o artefato faz e os critérios de aceitação especificados <ul style="list-style-type: none"> – além de testar requisitos funcionais, testa também a satisfação de requisitos não funcionais, inversos e de interface humana <p>Para evitar confusão de terminologia, selecionamos termos que correspondam ao objetivo de teste</p>		
	Mar 2017	Arndt von Staa © LES/DI/PUC-Rio	4

Tipos especiais de teste funcional?



- Especificações podem estar **erradas, incompletas, ou desatualizadas**
 - o **teste de aprovação** é uma variante do teste de aceitação e tem por objetivo descobrir discrepâncias entre o que o artefato faz e os **atuais** anseios **explícitos** e **implícitos** do usuário e do cliente, **independentemente da especificação**
 - leva em conta a possibilidade dos **anseios do usuário evoluírem** desde o início do desenvolvimento (especificação) até o momento do teste de aceitação
 - tornar disponível partes do sistema pode motivar o usuário a evoluir seus anseios
 - diferença:
 - **aceitação** → especificado antes de desenvolver, o teste verifica a conformidade com a especificação
 - **aprovação** (ou adequação) → realizada após estar disponível, o teste verifica a conformidade com os **atuais** anseios do usuário

Anseio é usado no sentido de desejo (pode ou não ser satisfeito) ou necessidade (deve ser satisfeito)

Visão abstrata de um teste funcional



- **Teste funcional**
 - Sempre existe um **contexto**, ex.
 - base de dados de teste
 - conteúdo desta base de dados
 - Podem existir **pré condições específicas** para o caso de teste
 - Existem **pós condições (oráculo)** que descrevem o resultado correto normal a ser produzido
 - o oráculo pode depender de condições de retorno
 - Podem existir **pós condições (oráculo)** que descrevem o resultado de término anormal a ser produzido, ex.
 - exceções
 - cancelamentos
 - É **irrelevante saber como o artefato transforma** entrada em resultados

Laboratório de Engenharia de Software

Critério: cobertura de elementos gráficos

Sistema Sis

Usuário

1

Senha

2

Digite os caracteres

Captcha

3

Login
Cancelar
Mudar senha
Esqueci senha

4
5
6
7

- Condições
 - Campo usuário:
 - 1,a : vazio
 - 1,b : igual a um existente
 - 1,c : igual a um existente, mas com 1 caractere a menos
 - 1,d : igual a um existente, mas com 1 caractere a mais
 - 1,e : máximo permitido + 50 caracteres
 - . . .

Mar 2017
Arndt von Staa © LES/DI/PUC-Rio
7

Laboratório de Engenharia de Software

Critério: cobertura de elementos gráficos

- Dado um esboço ou desenho da janela, crie um identificador único para cada elemento (*widget*) da interface gráfica
 - widgets: caixas, botões, seletores, atalhos, réguas . . .
 - o id é usualmente um número, ou um nome (*string*) bem curto
- Gere casos de teste úteis
 - leve em conta as especificidades de cada *widget*
 - utilize os critérios de valoração para definir as especificidades
 - para cada um dos casos de teste marque todos os itens por ele exercitados
- Gere a suíte de teste
 - gere diferentes casos até que todos os itens tenham sido marcados pelo menos uma vez, e todas as condições de valoração pertinentes tenham sido consideradas
 - verifique problemas de mascaramento e ambiguidade

Mar 2017
Arndt von Staa © LES/DI/PUC-Rio
8

Cobertura de elementos gráficos



- Especificidades de cada *widget*, ex:
 - list box – testar cada item da lista
 - *radio button* – testar cada alternativa
 - campo de entrada, testar:
 - vazio,
 - com 1 caractere,
 - com `dimCampo` caracteres,
 - com `dimCampo+1` caracteres
 - com `dimCampo+500` caracteres
 - barra de rolagem: seta, elevador, até o limite, além do limite
 - atalhos (*hot keys*)
 - botões – testar cada botão
 - existindo campos condicionais, testar cada resultado de condição. Caso seja condição composta: cada condição elementar

a dimensão do campo deve estar documentada → documentação técnica

um número “grande” qualquer

devem estar documentados

Exemplo de casos de teste : lista corrida



- Usuário vazio, Senha vazio, Captcha igual, Login
 - Campo usuário não existe
- Usuário vazio, Senha vazio, Captcha igual, Cancela
 - cancela acesso
- Usuário vazio, Senha vazio, Captcha igual, Mudar senha
 - Campo usuário deve conter no mínimo n1 caracteres
- Usuário vazio, Senha vazio, Captcha igual, Esqueci senha
 - Campo usuário deve conter no mínimo n1 caracteres
- Usuário existe, Senha vazio, Captcha igual, Login
 - Campo senha não existe
- Usuário existe mas digitou -1, Senha vazio, Captcha igual, Login
 - Usuário não está cadastrado
- . . .

Isso é uma boa resposta?

Isso é uma boa resposta?

- Mar 2017

LES

Usuário	vazio	s													
	existe		s				s		s	s					
	existe -1			s								s			
	existe +1				s								s		
	máximo +50					s									s
Senha	vazio														
	existe		s	s	s	s			s			s	s	s	s
	existe -1														
	existe +1														
	máximo +50														
	existe de outro						s								
Captcha	corretos	s	s	s	s	s	s		s	s	f	s	s	s	s
Botões	Login	s	s	s	s	s	s					s			
	Cancela							s							
	Troca								s					s	s
	Esqueci									s					
Ação	Autoriza		s												
	Erro de uso	s		s	s	s	s					s	s	s	s
	Troca senha								s						
	Nova senha									s					
	Cancela							s							

preenchida de
forma não
sistemática

— não deveria
testar com
máx +50?

Critério: Tabelas de decisão



- Contexto típico para o uso de tabelas de decisão:
 - tabelas de decisão têm por objetivo simplificar o entendimento de decisões compostas
 - cada combinação de decisões dispara um determinado conjunto de uma ou mais ações
 - teste: cada combinação de decisões corresponde a um determinado oráculo
- **Problema**: um número grande de condições aninhadas pode ser muito difícil de tratar
 - tabelas de decisão são exponenciais: n condições binárias resultam em 2^n possíveis casos de teste
 - chama-se isso de **explosão combinatória**
 - problema comum em testes sistemáticos

Tabelas de decisão: sintaxe



- Na tabela abaixo são identificadas n **decisões** e k **ações**
- Cada uma das L **colunas** define uma **combinação de condições** que, se satisfeitas, ativam uma ou mais ações
- Cada coluna corresponde a um **caso de teste**


	Col 1	Col 2	...	Col L
Decisão 1				
Decisão 2				

Decisão n				
Ação 1				
Ação 2				

Ação k				

Laboratório de Engenharia de Software

Tabelas de decisão




- Usualmente as tabelas utilizam **decisões** binárias
- As **condições** de uma decisão podem ser
 - S** ou **V** ou **T** verdadeiro
 - N** ou **F** ou **F** falso
 - indiferente (tanto faz se verdadeiro ou falso)
 - a decisão pode, ou não, ser avaliada
 - reduz o número de colunas, cada "-" corresponde a duas colunas
 - cuidado com ambiguidades
 - N/A** não se aplica
 - a decisão **não será avaliada** em virtude das condições definidas para as outras decisões
 - tratada como indiferente para fins de avaliação da tabela

Condição é o valor esperado de uma decisão

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

Laboratório de Engenharia de Software

Tabelas de decisão



- Decisões não binárias devem ser convertidas em uma coletânea de decisões binárias.
- Ex.
 - a relação $a \leq x < b$ pode ser convertida nas seguintes decisões, a e b são dados x é selecionado
 - $a - \varepsilon == x \Rightarrow$ não vale
 - $a == x \Rightarrow$ vale
 - $a + \varepsilon == x \Rightarrow$ vale
 - $b - \varepsilon == x \Rightarrow$ vale
 - $b == x \Rightarrow$ não vale
 - $b + \varepsilon == x \Rightarrow$ não vale

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

Exemplo



Validar se x está no intervalo int: $0 \leq x < 100$

O exemplo utiliza dados já valorados segundo as regras de valoração

$x < 100 \equiv x \leq 99$

$x == -1$	s	n	n	n	n	n	n	n
$x == 0$	-	s	n	n	n	n	n	n
$x == 1$	-	-	s	n	n	n	n	n
$x == 50$	-	-	-	s	n	n	n	n
$x == 98$	-	-	-	-	s	n	n	n
$x == 99$	-	-	-	-	-	s	n	n
$x == 100$	-	-	-	-	-	-	s	n
x vale		x	x	x	x	x		
x não vale	x						x	
impossível								x

por que?

Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

17

Tabelas de decisão: casos especiais



- Decisões **mutuamente exclusivas** → no máximo uma das condições poderá ser *Verdadeira*
 - n linhas (o número de condições) correspondem a 2^n colunas,
 - exemplo: *radio button*
 - exemplo: botões de seleção de ação botões clicáveis
 - exemplo: as três decisões mutuamente exclusivas $x < N$, $x == N$ e $x > N$ podem ser implementadas com duas decisões mutuamente exclusivas: $x < N$ e $x == N$
 - neste caso $n == 2$

mutuamente exclusivo
e obrigatório

mutuamente exclusivo
e não obrigatório

O símbolo $::=$ significa "definido por" ou "definido como"


Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

18

Laboratório de Engenharia de Software

Tabelas de decisão: casos especiais



- **Conjunto obrigatório** é um conjunto de 2 ou mais decisões no qual pelo menos uma deve resultar na condição V
 - exemplo: a decisão ternária $x < N$, $x == N$ e $x > N$, forma um conjunto obrigatório em que cada decisão é mutuamente exclusiva com as demais
 - três linhas correspondem a 2^n colunas
 - no caso $n == 3$


Mutuamente exclusivo e obrigatório

$x < N$	T	T	T	T	F	F	F	F
$x == N$	T	T	F	F	T	T	F	F
$x > N$	T	F	T	F	T	F	T	F
	imp	imp	imp	$x < N$	imp	$x == N$	$x > N$	imp

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

Laboratório de Engenharia de Software

Tabelas de decisão: casos especiais



- **mascamamento** ou **consequência de outra** decisão em um conjunto de decisões: ocorre se uma decisão resultar em verdadeiro e então as outras necessariamente resultarão em verdadeiro (ou falso).
 - exemplo: se particionarmos os valores de uma variável em 4 faixas, teremos ao todo as seguintes 6 condições binárias:
 - $x < V1$, $V1 \leq x$, $x < V2$, $V2 \leq x$, $x < V3$, $V3 \leq x$ em que $V1 < V2 < V3$
 - se $x < V1$ então, por transitividade, também $x < V2$
 - ou seja, $x < V1$ mascara a condição $x < V2$

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

Laboratório de Engenharia de Software

Tabelas de decisão: casos especiais

assuma: $n1 < n2 < n3$, mascaramento

$x < n1$	T	F	F	F
$x < n2$	—	T	F	F
$x < n3$	—	—	T	F
conta	4	2	1	1

Por que — ?
Por que F e não — ?

$x \geq n3$, por que?

Como adicionar o critério de valoração ao conjunto de condições?

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

Laboratório de Engenharia de Software

Tabelas de decisão: casos especiais


$x < n1$	T	F	F	F	F	F	F
$x == n1$	—	T	F	F	F	F	F
$x < n2$	—	—	T	F	F	F	F
$x == n2$	—	—	—	T	F	F	F
$x < n3$	—	—	—	—	T	F	F
$x == n3$	—	—	—	—	—	T	F
conta	32	16	8	4	2	1	1

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

Laboratório de Engenharia de Software

Tabelas de decisão: casos especiais

- Verificação da **ambiguidade** (redundância, inconsistência)
 - uma coluna X é ambígua com relação a outra Y , se existirem condições que permitam selecionar simultaneamente as duas



$x < n1$	T	—	—	F
$x < n2$	—	T	—	F
$x < n3$	—	—	T	F
conta	4	4	4	1

Se o total da contagem der mais do que 2^n existem colunas ambíguas

Mas para totais menores também podem existir ambiguidades!


13 deveria ser 8

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

Laboratório de Engenharia de Software

Tabelas de decisão: casos especiais

- Verificação da **completeza** em tabelas com n **decisões binárias**:
 - no quadro completo devem existir $L == 2^n$ colunas
 - cada condição indiferente conta por duas alternativas.
 - as alternativas em uma coluna são multiplicadas
 - uma forma simples é criar tabelas com campos indiferentes e N/A somente à esquerda



$x < n1$	T	F	F	F
$x < n2$	—	T	F	F
$x < n3$	—	—	T	F
conta	4	2	1	1

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

[illegible]


- Qual seria o número total de colunas?
- Qual seria o número aproximado de colunas, levando em conta as restrições dos grupos

- Quais são os grupos e quais são as restrições que governam cada grupo?
- Após preencher, a tabela está completa? Não ambígua?
- Todos grupos capturam corretamente as características do(s) *widget(s)*?

APÊNDICE

Quem são usuário e cliente?

recordação



Laboratório de Engenharia de Software

- **Usuário** (*user*) pode ser
 - pessoa interessada no **serviço prestado** pelo artefato sob teste
 - pessoa interessada em **manter** o artefato sob teste
 - pessoa interessada em **por em operação** o artefato sob teste
 - pessoa interessada em **compor** o artefato sob teste com outros
 - outro artefato (software) com o qual o artefato sob teste **interagirá**
 - equipamento (máquina) com o qual o artefato sob teste **interagirá**
 - ...
- **Cliente** (*customer*) é a pessoa ou organização
 - que disponibiliza recursos para a aquisição, desenvolvimento ou manutenção do artefato
 - interessada na redução de custos e riscos incorridos pelos processos
 - interessada na viabilização de um serviço impossível de ser realizado sem o emprego de sistemas intensivos em software
 - ...

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

Referências bibliográficas



Laboratório de Engenharia de Software

- Caldeira, L.R.N.; *Geração semi-automática de massas de testes funcionais a partir da composição de casos de uso e tabelas de decisão*; Dissertação de Mestrado, DI/PUC-Rio; 17/ago/2010
- Delamaro, M.E.; Maldonado, J.C.; Jino, M.; *Introdução ao Teste de Software*; Rio de Janeiro, RJ: Elsevier / Campus; 2007
- Kaner, C.; Falk, J.; Nguyen, H.Q.; *Testing Computer Software*; 2nd edition; London: Thomson; 1993
- Lachtermacher, L.; : *Geração Automática de Dados de Teste através de Tabelas de Decisão*; Dissertação de Mestrado, DI/PUC-Rio; 5/3/2010
- Mousavi, M.; *Decision Table-Based Testing*; Lecture notes; Eindhoven University of Technology; 2008
- Myers, G.J.; *The Art of Software Testing*; 2nd edition; New York: John Wiley & Sons; 2004

Mar 2017
Arndt von Staa © LES/DI/PUC-Rio
28

Laboratório de Engenharia de Software

LES

FIM

Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

29