




Laboratório de Engenharia de Software

## Teste Funcional 3

Arndt von Staa  
Departamento de Informática  
PUC-Rio  
Março 2017

## Especificação




Laboratório de Engenharia de Software

- Objetivo desse módulo
  - Detalhar o uso de testes baseados em comportamento como um instrumento de especificação através de exemplos. Apresentar uma modalidade de criação de casos de teste a partir de casos de uso
- Justificativa
  - Casos de uso são utilizados para especificar sistemas
  - Especificações devem ser verificáveis
  - É desejável que, além de serem verificáveis, seja possível gerar os casos de teste diretamente a partir das especificações, mesmo que utilizando técnicas semi-automatizadas
  - É desejável ser capaz de gerar casos de teste baseados em comportamento como um instrumento de controle da qualidade de casos de uso.

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

Laboratório de Engenharia de Software

## Motivação




- Ideal
  - Usar exemplos para revisar ou inspecionar a especificação
  - Ser capaz de executar automaticamente os testes de aceitação
  - Ser capaz de gerar automaticamente os testes de aceitação
- Quanto mais cedo forem criados os casos de teste, melhor
  - idealmente deveriam ser criados junto com a especificação
    - desenvolvimento Dirigido por Teste de Aceitação  
*ATDD Acceptance Test Driven Development*
  - a especificação pode ser formada ou complementada por uma série de exemplos
    - cada exemplo passa a ser um caso de teste
      - vantagem: é possível examinar a **adequação** a partir dos exemplos
      - vantagem: é possível controlar **ambiguidade** na especificação através de **oráculos bem definidos**
      - desvantagem: risco de estar **incompleto, incorreto, inconsistente**

Adzic, G.; *Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing*;

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


Laboratório de Engenharia de Software


## Terminologia



- **Cenário** – o espaço real ou virtual em que a história se passa.
  - No teatro o conjunto de elementos que decoram o palco (wikipedia)
  - Em testes: as pré condições de um caso de teste
    - **contexto** – dados persistentes ou não e que serão repetidamente usados em diversos casos
    - **pré condições** – dados específicos para determinado caso de teste
- **Cena** – sequência de ações executadas por um caso de teste
  - ex. um caminho no grafo de fluxo de um caso de uso
- **Ação** – operação indivisível
  - atuação elementar do usuário
  - processamento **bem delimitado** do artefato sob teste,
    - frequentemente indica a alteração do estado do processamento


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

Laboratório de Engenharia de Software	<b>Mais terminologia</b>	
	<ul style="list-style-type: none"> <li>• <b>Criação</b> de caso de teste – sequência de ações humanas que culmina com a redação de um caso de teste útil             <ul style="list-style-type: none"> <li>– o caso de teste útil pode ser um caso de teste dirigido por comportamento</li> </ul> </li> <li>• <b>Geração</b> de caso de teste – sequência de ações desempenhadas por um sistema e que culmina com a redação de um caso de teste útil</li> </ul>	
	Mar 2017	Arndt von Staa © LES/DI/PUC-Rio 5

Laboratório de Engenharia de Software	<b>Teste dirigido por comportamento</b>	
	<ul style="list-style-type: none"> <li>• O <b>teste dirigido por comportamento</b> (<i>behaviour driven testing</i>) baseia-se na criação de casos de teste a partir de cenas de uso             <ul style="list-style-type: none"> <li>– uma forma alternativa é: gerar casos de teste diretamente a partir de cenas de uso</li> </ul> </li> <li>• O objetivo dessa forma de criação de casos de teste é produzir simultaneamente:             <ul style="list-style-type: none"> <li>– uma redação inteligível pelos interessados</li> <li>– uma redação suficientemente precisa para permitir o desenvolvimento de uma funcionalidade</li> </ul> </li> </ul> <p>North, D.; Behavior Driven Development (BDD) <a href="https://dannorth.net/introducing-bdd/">https://dannorth.net/introducing-bdd/</a></p>	
	Mar 2017	Arndt von Staa © LES/DI/PUC-Rio 6

Laboratório de Engenharia de Software

## Teste dirigido por comportamento (recordação)



- Existem várias formas de redigir historietas
- Historieta (user story)
 

Como Administrador do sistema

Eu quero ser capaz de gerenciar os dados dos usuários


Para poder manter o sistema atualizado

  - incompleto e ambíguo:
    - o que entendemos por gerenciar dados do usuário?
    - o que entendemos por manter o sistema atualizado?
  - solução: exemplos precisos para cada ação de gerenciar

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

Laboratório de Engenharia de Software

## Especificação usando exemplos



- Uma cena
 

Dado que estou autenticado como Administrador

Quando eu clicar no botão Listar usuários

Então será exibida a lista de todos os usuários cadastrados

Quando eu selecionar nesta lista o usuário José

E clicar no botão Apagar

Então será reexibida a lista dos usuários cadastrados

E esta lista não conterá o usuário José
- Que outras cenas você proporia?

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

Laboratório de Engenharia de Software

## Teste dirigido por comportamento


- Geração de caso teste automatizado a partir de um exemplo de comportamento

```

Scenario: Admin successfully creates user

Given I am logged in with user "admin" and password "admin"
When I choose to create a new user and
When I enter name "Maria" and
When I enter email "maria@gmail.com" and
When I enter login "maria" and
When I enter password "maria123" and
When I click save button
Then I should see the user "Maria" in the list of users
          
```

- Basta ver o usuário Maria? Ou seria necessário ver também os demais dados?



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

Laboratório de Engenharia de Software

## Teste dirigido por comportamento

```

@When( "I click save button" )
public void clickSaveButton() throws InterruptedException
{
    this.selenium.click( "bar" );
    Thread.sleep( 5000 );
}


@When( "I enter email \"$email\" and" )
public void SetUserEmail( final String email )
{
    this.selenium.type( "email", email );
}

@When( "I enter login \"$login\" and" )
public void setUserLogin( final String login )
{
    this.selenium.type( "chave", login );
}

@When( "I enter name \"$name\" and" )
public void SetUserName( final String name )
{
    this.selenium.type( "nome", name );
}
          
```

Código gerado para Selenium

incompleto



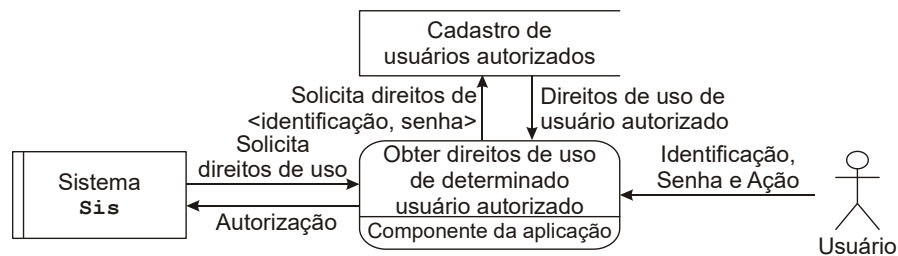
Mar 2017
Arndt von Staa © LES/DI/PUC-Rio
10

## Criação de casos de teste a partir de casos de uso



- Identificar o **cenário**
  - o **contexto necessário** para um conjunto de casos de teste
  - as **pré-condições necessárias** para poder realizar o teste
- Criar o conjunto de **cenas**
  - um caso de uso pode ser representado por um grafo, cada caminho nesse grafo é uma cena
  - identificar os **caminhos** existentes, usualmente em um grafo
    - um caminho começa na origem do grafo e registra cada elemento (vértice) visitado. Possivelmente visita diversas vezes um ou mais vértices
  - cada caminho narra a sequência de ações e os resultados observáveis
- Identificar as **cenas normais**
  - cenas normais terminam em um retorno esperado
- Identificar as **cenas anormais**
  - cenas anormais terminam em um retorno não esperado, especificado pela garantia mínima
    - por exemplo exceções próprias ou gradas em um outro método

## Ambiente para o teste componente login




- Para **poder testar** o componente, é necessário criar uma **armadura de teste que simule o comportamento** dos possíveis (muitos...) sistemas Sis
- A **interface** do componente deve estar **especificada precisamente**
- A interface da armadura com o componente deve obedecer exatamente à especificação
  - dessa forma qualquer sistema Sis que obedeça a essa interface poderá confiar no funcionamento do componente

Laboratório de Engenharia de Software

## Contexto

- Para poder ser testado, o sistema **TesteSis** precisa estar vinculado a uma base de dados contendo os dados dos usuários cadastrados a serem utilizados durante os testes, ex:
  - O sistema **TesteSis** deve estar vinculado à base de dados de usuários **TesteSis.usuários**
  - A base de dados **TesteSis.usuarios** deve estar inicializada e criptografada com a senha de teste **xpto###**
  - **TesteSis.usuarios** , contém:
    - <idUserio: joaoSilva , Senha: joao#### , direitos: {a,b,c}>
    - <idUserio: mariaSa , Senha: #maria# , direitos: {c,d,e}>
    - <idUserio: joseGomes , Senha: joao#### , direitos: {a,b,e}>




Mar 2017
Arndt von Staa © LES/DI/PUC-Rio
13

Laboratório de Engenharia de Software

## Contexto

- O ideal seria a base de dados ser gerada para cada caso de teste
  - assegura que o teste não falhe em virtude de mudanças inesperadas no conteúdo da base de dados
  - assegura que o teste não corrompa a base de dados para testes subsequentes
  - porém aumenta  **muito**  o tempo de execução do teste
    - para reduzir o custo é comum gerar a base de dados para um conjunto de casos de teste que se tem certeza não se interferam mutuamente
  - pode ser realizado com **DBUnit**, ou um módulo (*mock object*) especificamente projetado para esse fim



Mar 2017
Arndt von Staa © LES/DI/PUC-Rio
14

## Parêntesis



- Use uma **terminologia padrão** para redigir o roteiro.
- Os termos devem estar associados à natureza do *widget*
  - **digitar** entrada de dados em campo de texto, ou string
  - **clicar** “pressionar” um botão
  - **selecionar** escolher uma das opções de “radio button”
  - **marcar** selecionar uma das opções de “check box”
  - **escolher** selecionar uma das opções de “list box”
  - **escolher vários** selecionar duas ou mais das opções de “list box”
  - **verificar** aplica um oráculo de controle intermediário  
(na realidade não é um *widget*...)
  - ...

Algumas ferramentas de teste padronizam terminologias para uso próprio

## Cena do login normal (caso de teste semântico)



- O *driver* de teste **ativa** o componente
  - os campos de entrada **devem estar** em branco
  - o *captcha* **deve estar** gerado
- O usuário **digita** *idUsuário* e *senha* de usuário autorizado
- O usuário **digita** corretamente o *captcha*
- O componente verifica que o *captcha* está válido
- O usuário **clica** *Login*
- O componente verifica que  $\langle idUsuário, senha \rangle$  vale
- O componente **retorna** ao *driver* de teste:  $\{ usuário autorizado, direitos de uso \}$  correspondentes ao usuário selecionado



Laboratório de Engenharia de Software

## Cena login normal (caso de teste útil)

- Assegure que o driver de teste esteja vinculado ao cadastro `TesteSis.usuarios`, criptografado com senha `XPTO###`
- Usando o *driver* de teste, ativar o Controle de Acesso
- Após a janela abrir
  - verificar se os campos estão vazios
  - digitar `idUsuário = joao.silva`
  - digitar a `senha = joao####`
  - verificar se o campo senha não exibe os caracteres digitados
  - digitar corretamente o `captcha`
  - clicar o botão `Login`
  - verificar se a janela fechou
- Usando o *driver* de teste, verificar se os dados retornados são `{Autorizado, { a,b,c }}`
- Usando o *driver* de teste, verificar todas as condições de término
 

é necessário especificar como o conjunto retornado é codificado.

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

Laboratório de Engenharia de Software

## Caso de teste do login normal formulário


<b>IDENTIFICADOR DO CASO DE TESTE:</b> Login001		Explicitando as ações do testador
<b>DESCRIÇÃO:</b> O usuário realiza um Login bem sucedido		
<b>PRE-CONDICOES:</b> <ul style="list-style-type: none"> <li>Sistema TesteSis (i.e. o <i>driver</i> de teste) ativa controle de acesso com o cadastro <code>TesteSis.usuarios</code></li> <li>O cadastro está criptografado com a senha de teste <code>XPTO###</code></li> <li><span style="color: red;">Controle de acesso</span> abriu a janela de login</li> </ul>	<b>AÇÕES DO USUÁRIO:</b> <ul style="list-style-type: none"> <li><span style="color: red;">Verificar</span> se identificação está vazia</li> <li><span style="color: red;">Verificar</span> se senha está vazia</li> <li><span style="color: red;">Verificar</span> se o <code>captcha</code> está gerado</li> <li><span style="color: red;">Digitar</span> a identificação: <code>joaoSilva</code></li> <li><span style="color: red;">Digitar</span> a senha: <code>joao####</code></li> <li><span style="color: red;">Verificar</span> se campo senha contém somente '*'</li> <li><span style="color: red;">Digitar</span> exatamente o <code>captcha</code></li> <li><span style="color: red;">Clicar</span> "Login"</li> </ul>	<b>POS-CONDICOES:</b> <ul style="list-style-type: none"> <li><span style="color: red;">Controle de acesso</span> fechou a janela de login</li> <li><span style="color: red;">Verificar</span> se TesteSis recebeu o resultado: <code>{autorizado, {a,b,c}}</code></li> <li><span style="color: red;">Verificar</span> se foi satisfeito o requisito de término:                     <ul style="list-style-type: none"> <li>espaços de dados usados obliterados</li> </ul> </li> </ul>
<b>CRITÉRIO DE SUCESSO:</b>		

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

Laboratório de Engenharia de Software

## Cena de captcha incorreto

- Os campos de entrada estão em branco
- O captcha está gerado
- O usuário digita *idUsuário* e *senha* de usuário autorizado
- O usuário digita incorretamente o *captcha*
- O usuário clica Login
- O sistema observa o erro do *captcha*
- Se for o primeiro ao terceiro erro inclusive
  - O sistema emite a mensagem de erro “Erro de digitação”
  - Controle de acesso retorna à aquisição de dados
- Senha:
  - Controle de acesso emite a mensagem de erro “Usuário não autorizado, processamento cancelado.”
  - Retorna {*usuário não autorizado* , *direitos de uso vazio*}




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

Laboratório de Engenharia de Software

## Cena cancelar com dados válidos


- Assegure que esteja em uso o cadastro  
`TesteSis.usuarios` , criptografado com senha `xpto###`
- Usando o *driver* de teste, ativar o Controle de Acesso
- Após a janela abrir
  - **verificar** se os campos estão vazios
  - digitar `idUsuário = joao.silva`
  - digitar a senha correta = `joao####`
  - digitar corretamente o captcha
  - clicar o botão **Cancelar**
  - **verificar** se a janela fechou
- Usando o *driver* de teste, **verificar** se os dados retornados são  
{ `cancelado` , `nulo` }
- Usando o *driver* de teste, verificar todas as condições de término



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

Laboratório de Engenharia de Software

## Cena senha fornecida errada uma só vez




- Assegure que esteja em uso o cadastro  
`TesteSis.usuarios` , criptografado com senha `xpto###`
- Usando o *driver* de teste, ativar o Controle de Acesso
- Após a janela abrir o usuário deve
  - digitar `idUsuario = joao.silva`
  - digitar a senha incorreta = `joao###`
  - digitar corretamente o captcha
  - clicar o botão *Login*
  - observar que recebeu a mensagem “Usuário não conhecido”
  - clicar o botão *OK* da mensagem
  - verificar se retornou à janela de dados, com os campos usuário e senha apagados e captcha diferente da vez anterior
  - digitar `idUsuario = joao.silva`
  - digitar a senha correta = `joao####`
  - digitar corretamente o captcha
  - clicar o botão *Login*
  - verificar se a janela fechou
- Usando o *driver* de teste, verificar se os dados retornados são  
`{ autorizado , { a,b,c } }`
- Usando o *driver* de teste, verificar todas as condições de término

Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

21

Laboratório de Engenharia de Software



O resto fica para exercício 😊

Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

22

## Outras cenas



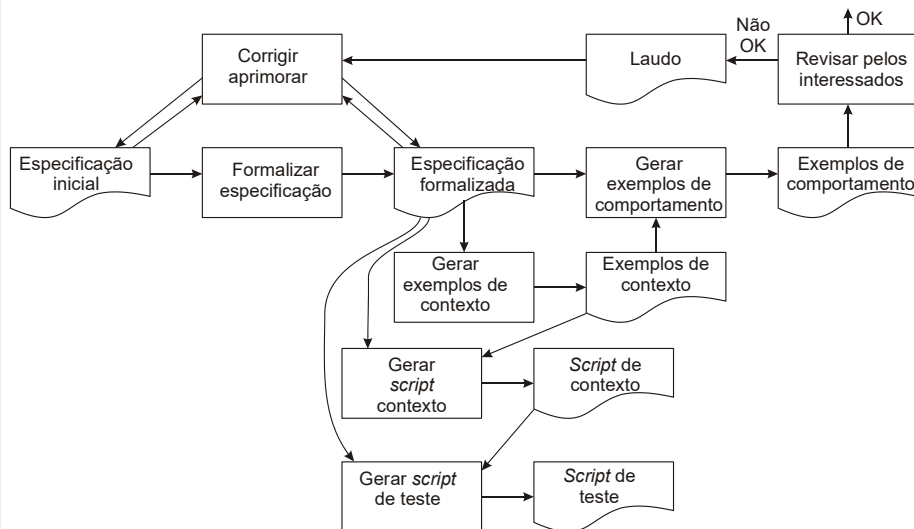
- Continua-se criando cenas de teste de forma similar ao que foi feito até agora
- Problema: como saber se foram criadas todas as cenas (relevantes)?
- Solução:
  - criar uma máquina de estados (próxima aula)
  - criar a gramática que descreve o conjunto de todos os caminhos possíveis (a ser visto nas aulas de teste estrutural)
  - criar caminhos segundo uma regra de completeza (a ser visto nas aulas de teste estrutural)

Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

23

## Possível automação



Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

24

Laboratório de Engenharia de Software

# APÊNDICE

LES

Mar 2017
Arndt von Staa © LES/DI/PUC-Rio
25 / 31

Laboratório de Engenharia de Software

## Como automatizar?

- Pode-se gerar um script com uma ferramenta de *capture and replay* (SQUISH)

1. O programa deve ser ativado.
2. O usuário deve selecionar o nome *Thiago* no combobox *ComboAluno*.
3. O usuário deve ativar o botão *CalcularMedia*.
4. O programa deve exibir o diálogo de calculo de média.
5. O usuário deve preencher o campo *P1* com o valor 7.3.
6. O usuário deve preencher o campo *P2* com o valor 6.7.
7. O usuário deve preencher o campo *P3* com o valor 4.1.
8. O usuário deve preencher o campo *P4* com o valor 8.5.
9. O usuário deve ativar o botão *Calcular*.
10. O programa deve exibir o valor 6.65 no campo *Media*.
11. O usuário deve ativar o botão *Aceitar*.
12. O programa deve fechar o diálogo.
13. O usuário deve ativar o botão *Fechar*.
14. O programa deve terminar.

LES

Araújo, T.P.; Staa, A.v.; *Um Método Baseado em Comportamento com Foco no Desenvolvimento de Aplicações Baseadas em Interfaces Gráficas*;

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

## Como automatizar?



Usando a ferramenta SQUISH é produzido o script na linguagem Python

```
1. def main():
2.     waitForObjectItem(":groupBox1.comboBoxName_QComboBox", "Thiago")
3.     clickItem(":groupBox1.comboBoxName_QComboBox", "Thiago", 51, 7, 1, Qt.LeftButton)
4.     waitForObject(":groupBox1.Calcular média_QPushButton")
5.     clickButton(":groupBox1.Calcular média_QPushButton")
6.     waitForObject(":Calcular Média.lineEditP1_QLineEdit")
7.     mouseClick(":Calcular Média.lineEditP1_QLineEdit", 49, 13, 1, Qt.LeftButton)
8.     waitForObject(":Calcular Média.lineEditP1_QLineEdit")
9.     type(":Calcular Média.lineEditP1_QLineEdit", "7.3")
10.    waitForObject(":Calcular Média.lineEditP2_QLineEdit")
11.    mouseClick(":Calcular Média.lineEditP2_QLineEdit", 21, 8, 1, Qt.LeftButton)
12.    waitForObject(":Calcular Média.lineEditP2_QLineEdit")
13.    type(":Calcular Média.lineEditP2_QLineEdit", "6.7")
14.    waitForObject(":Calcular Média.lineEditP3_QLineEdit")
15.    mouseClick(":Calcular Média.lineEditP3_QLineEdit", 37, 10, 1, Qt.LeftButton)
16.    waitForObject(":Calcular Média.lineEditP3_QLineEdit")
17.    type(":Calcular Média.lineEditP3_QLineEdit", "4.1")
18.    waitForObject(":Calcular Média.lineEditP4_QLineEdit")
19.    mouseClick(":Calcular Média.lineEditP4_QLineEdit", 42, 10, 1, Qt.LeftButton)
20.    waitForObject(":Calcular Média.lineEditP4_QLineEdit")
21.    type(":Calcular Média.lineEditP4_QLineEdit", "8.5")
22.    waitForObject(":Calcular Média.Calcular_QPushButton")
23.    clickButton(":Calcular Média.Calcular_QPushButton")
24.    waitForObject(":Calcular Média.Aceitar_QPushButton")
25.    clickButton(":Calcular Média.Aceitar_QPushButton")
26.    waitForObject(":GBDD Example.Fechar_QPushButton")
27.    clickButton(":GBDD Example.Fechar_QPushButton")
```

Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

27

## Geração a partir de casos de uso



- A ferramenta no [site https://github.com/funtester/](https://github.com/funtester/) permite gerar conjuntos de casos de teste para casos de uso
  - permite vincular o gerador a uma base de dados de teste
  - reduz o número de fluxos alternativos através do uso de regras de negócio estabelecendo a condição a ser feita e a mensagem que deveria ser gerada caso a regra seja violada
  - trata critérios de valoração
  - infelizmente ainda está incompleta, foi criada como protótipo


Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

28

Laboratório de Engenharia de Software

## Uso de dados variáveis



- Ao invés de dados constantes pode-se utilizar variáveis
  - Assegurar que o Cadastro de usuários contenha vários `{idUsuario1, senha1}` → `{conjunto1}` a serem selecionados aleatoriamente
  - Usando o *driver* de teste, ativar o componente Controle de Acesso solicitando autorização de uso
  - Após a janela abrir o usuário deve
    - fornecer idUsuário = `idUsuario1`
    - fornecer senha = `senha1`
    - digitar o captcha
    - selecionar o botão `idLogin`
    - verificar se a janela fechou
  - No *driver* de teste verificar se o conjunto de direitos de uso retornado é `{Login, conjunto1}`

Como resolver esse?

Varia a cada vez que se pode fornecer dados.


Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

29

Laboratório de Engenharia de Software

## Uso de dados variáveis a cada ativação



- Solução 1 – fixar os dados
  - durante o teste gerar sempre o mesmo captcha
  - precisa alterar o cenário
  - precisa testar a geração e verificação do captcha em separado
    - ruim: a geração do captcha poderia interferir na criação da janela


Mar 2017

Arndt von Staa © LES/DI/PUC-Rio

30

Laboratório de Engenharia de Software

## Uso de dados variáveis a cada ativação




- Solução 2 – uso de uma função “call back” (instrumentação para teste automatizado)
  - a instrumentação é incluída por compilação condicional sse compilado para teste (`_DEBUG`)
  - instrumenta-se o gerador do captcha para
    - guardar os caracteres em alguma variável global
    - disponibilizar uma função que fornece o captcha guardado de modo que se possa simular a sua digitação
    - disponibilizar uma função que verifica se o captcha é diferente a cada nova geração
  - instrumenta-se o “digitador” do captcha para
    - buscar os caracteres guardados
    - simular a digitação

Mar 2017
Arndt von Staa © LES/DI/PUC-Rio
31

Laboratório de Engenharia de Software

## Uso de dados variáveis



- Vantagens do uso de dados variáveis
  - pode-se gerar automaticamente dados de teste
    - por exemplo a partir de uma tabela de decisão
  - a automação da valoração dos casos de teste reduz os custos de criação dos testes
  - a geração de um grande número de dados aleatórios aumenta a chance de se criar “sequências de uso extensas”, ou pouco comuns, ou até mesmo incoerentes ao se considerar o domínio da aplicação
    - teste do macaco, ou do gato?
  - a geração de dados aleatórios aumenta a chance de exercitar caminhos que ainda não foram exercitados
    - a cada ativação do teste usar uma outra semente para o gerador de números aleatórios: usualmente o relógio

Mar 2017
Arndt von Staa © LES/DI/PUC-Rio
32



- Mar 2017

Mar 2017