

Referências Bibliográficas

ATALLAH, M.J., 2000. *Algorithms and Theory of Computation Handbook*. CRC Press.

BARNETT, M. et al., 2003. *Model-Based Testing with AsmL.NET*. Redmond, USA: Microsoft Research.

BEIZER, B., 1990. *Software Testing Techniques, 2nd edition*. Van Nostrand Reinhold Co.

BENDER, R., 1996. *Proposed Software Evaluation and Test KPA n. 4*. [Online] Available at: <http://www.softtest.com/pages/kpabai.htm>.

BERTOLINI, C. & MOTA, A., 2010. *A Framework for GUI Testing based on Use Case Design*. Recife-PE, Brazil: Federal University of Pernambuco.

BINDER, R.V., 2000. *Testing Object-Oriented Systems: Models, Patterns and Tools*. Addison-Wesley.

BRAY, T. et al., 2008. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. [Online] Available at: <http://www.w3.org/TR/REC-xml/>.

BRIAND, L.C., Labiche, Y. & Hea, S., 2008. Automating regression test selection based on UML designs. *Information and Software Technology*, n. 51.

BRITISH STANDARD 7925-1, 1998. *Software Testing: Vocabulary*.

CALDEIRA, L.R.N., 2010. *Geração semi-automática de massas de testes funcionais a partir da composição de casos de uso e tabelas de decisão*. Rio de Janeiro, Brasil: PUC-Rio.

CHEN, M. et al., 2007. *UML Activity Diagram-Based Automatic Test Case Generation For Java Programs*. Nanjing, China: Nanjing University.

CHEN, J. & SUBRAMANIAM, S., 2002. Specification-based Testing for GUI-based Applications. *Software Quality Journal* n.10, pp.205–24.

CHERNONOZHKIN, S.K., 2000. *Automated Test Generation and Static Analysis*. Novosibirsk, Russia: Ershov Institute of Information Systems.

COAD, P., LUCA, J.d. & LEFEBVRE, E., 1999. *Java Modeling In Color With UML: Enterprise Components and Process*. HeptaBooks.

COCKBURN, A., 2000. *Writing Effective Use Cases*. Addison-Wesley.

CROCKFORD, D., 2006. JSON: The Fat-Free Alternative to XML. In *XML 2006*. Boston, 2006.

CROSBY, P.B., 1979. *Quality is Free*. New-York: McGraw-Hill.

DENGER, C. & MORA, M.M., 2003. *Test Cases Derived from Requirement Specifications*. Fraunhofer IESE Report.

DÍAS, I., LOSAVIO, F., MATTEO, A. & PASTOR, O., 2004. A Specification Pattern for Use Cases. *Information & Management*, 41, pp.961-75.

DINH-TRONG, T., 2004. *A Systematic Approach to Testing UML Design Models*. Fort Collins, Colorado: Colorado State University.

EL-ATTAR, M. & MILLER, J., 2010. Developing comprehensive acceptance tests from use cases and robustness diagrams. *Requirements Engineering*, vol. 15, pp.285-306.

ESCALONA, M.J. et al., 2011. An overview on test generation from functional requirements. *The Journal of Systems and Software*, n. 84, pp.1379–93.

FOWLER, M., 2004. *Inversion of Control Containers and the Dependency Injection pattern*. [Online] Available at: <http://martinfowler.com/articles/injection.html>.

FOWLER, M. & BECK, K., 1999. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley.

FOWLER, M. & EVANS, E., 2005. *FluentInterface*. [Online] Available at: <http://martinfowler.com/bliki/FluentInterface.html>.

GAMMA, E., Helm, R., Johnson, R. & Vlissides, J., 1994. *Design Patterns*. Addison-Wesley.

GERSTING, J.L., 1993. *Fundamentos Matemáticos para Ciência da Computação - 3a ed.* LTC.

GUTIÉRREZ, J.J., 2005. *Generación de pruebas de sistema a partir de la especificación funcional*. Sevilha, Espanha: Universidad de Sevilla.

GUTIÉRREZ, J.J., ESCALONA, M.J., MEJÍAS, M. & TORRES, J., 2005. *Analysis of proposals to generations of system test cases from system requirements*. Sevilla, Spain: University of Sevilla.

GUTIÉRREZ, J.J., ESCALONA, M.J., MEJÍAS, M. & TORRES, J., 2010. *Generation of test cases from functional requirements - A survey*. University of Seville.

GUTIÉRREZ, J.J. et al., 2008. *A Case Study for Generating Test Cases from Use Cases*. Sevilha, Espanha: University of Sevilla.

HASSAN, H.A. & YOUSIF, Z.E., 2010. Generating test cases for platform independent model by using use case model. *International Journal of Engineering Science and Technology*, vol. 2.

HEUMANN, J., 2001. Generating Tests from Use Cases. *The Rational Edge-zine*.

IM, K., Im, T. & McGregor, J.D., 2008. Automating Test Case Definition Using a Domain Specific Language. *ACM Proceedings on Software Engineering*.

INTERNATIONAL INSTITUTE OF BUSINESS ANALYSIS, 2009. *A Guide to the Business Analysis Body of Knowledge*. IIBA.

ISABELLA, A. & RETNA, J.E., 2012. Study paper on test case generation for GUI based testing. *International Journal of Software Engineering & Applications (IJSEA)*, Vol.3, No.1, January.

JAYGARL, H., 2010. *Capture-based automated test input generation*. Ames, Iowa: Iowa State University.

JIANG, M. & DING, Z., 2011. *Automation of test case generation from textual use cases*. Hangzhou, China: Zhejiang Sci-Tech University.

KAHN, A.B., 1962. Topological sorting of large networks. *Communications of the ACM* 5, November. pp.558–62.

KASSEL, N.W., 2006. *An approach to automate test case generation from structured use cases*. Clemson University.

LABICHE, Y. & BRIAND, L., 2002. *A UML-Based Approach to System Testing*. Ottawa, Canada: Carleton University.

MAGALHÃES, J.A.P., 2009. *Recovery Oriented Software*. Rio de Janeiro: PUC-Rio. Tese de Doutorado.

MALDONADO, J.C., Sugeta, T., Masiero, P.C. & Fabbri, S.C.P.F., 1999. *Mutation Testing Applied to Validate Specifications Based on Statecharts*. São Carlos, São Paulo - Brazil: Universidade Federal de São Carlos / Universidade de São Paulo.

MARTIN, R.C., 2000. *Design Principles and Design Patterns*. [Online] Available at: http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf [Accessed 10 abril 2010].

MCCABE, T.J., 1976. A Complexity Measure. *IEEE Transactions on Software Engineering* n.4, December. pp.308-20.

MEMON, A., Nagarajan, A. & Xie, Q., 2003. *Automating Regression Testing for Evolving GUI Software*. Maryland, USA: University of Maryland.

MEMON, A.M., Soffa, M.L. & Pollack, M.E., 2001. *Coverage Criteria for GUI Testing*. Pittsburgh, PA: University of Pittsburgh / University of Michigan.

MEMON, A.M. & XIE, Q., 2005. Studying the Fault-Detection Effectiveness of GUI Test Cases for Rapidly Evolving Software. *IEEE Transactions on Software Engineering*, vol. 31, n. 10, October.

MEMON, A.M. & XIE, Q., 2007. Designing and comparing automated test oracles for GUI-based software applications. *ACM Transactions on Software Engineering Methodologies*, n. 16, article 4, February.

MESZAROS, G., 2007. *xUnit Test Patterns: Refactoring Test Code*. Addison-Wesley.

MEYER, B. & BAUDOUIN, C., 1980. *Méthodes de programmation*. Eyrolles.

MEYERS, G.J., 1979. *The Art of Software Testing*. New York: John Wiley & Sons.

MILER, K.W. et al., 1992. Estimating the probability of failure when testing reveals no failures. *IEEE Transactions on Software Engineering*, n. 18, pp.33-43.

MOLLER, A., 2010. *Finite-State Automata and Regular Expressions for Java*. [Online] Available at: <http://www.brics.dk/automaton/>.

NAVARRO, P.L.M., Ruiz, D.S. & Pérez, G.M., 2010. *A Proposal for Automatic Testing of GUIs Based on Annotated Use Cases*. Murcia, Spain: University of Murcia.

NEBUT, C., Fleurey, F., Le Traon, Y. & Jézéquel, J.-M., 2006. Automatic Test Generation: a use case driven approach. *IEEE Transactions on Software Engineering*, vol. 32, no. 3.

NIST, 2002. *The Economic Impacts of Inadequate Infrastructure for Software Testing*. NIST Planning Report 02-3.

OBJECT MANAGEMENT GROUP, 1997. *UML*. [Online] Available at: <http://www.omg.org/spec/UML/>.

OBJECT MANAGEMENT GROUP, 2003. *OCL*. [Online] Available at: <http://www.omg.org/spec/OCL/>.

OKUM, V., 2004. *Specification mutation for test generation and analysis*. Maryland, USA: University of Maryland.

PESSOA, M.B., 2011. *Geração e execução automática de scripts de teste para aplicações web a partir de casos de uso direcionados por comportamento*. Dissertação de Mestrado. Rio de Janeiro, Brasil: PUC-Rio.

RAMAKRISHNAIAH, T., 1999. *Automatic generation of test cases and anticipated test outcome based on a tabular design specification*. Montreal, Quebec, Canada: Department of Computer Science - Concordia University.

RAN, L. et al., 2008. Building test cases and oracles to automate the testing of web database applications. *Information and Software Technology*, n.51, pp.460-77.

RYSER, J. & GLINZ, M., 1999. A Scenario-Based Approach to Validating and Testing Software Systems Using Statecharts. In *12th International Conference on Software and Systems Engineering and their Applications*. Paris, France, 1999.

SAMUEL, P., Mall, R. & Bothra, A.K., 2007. Automatic test case generation using unified modeling language (UML) state diagrams. *The Institution of Engineering and Technology*, vol. 2, n. 2, pp.79-93.

SINHA, A., 2005. *Domain specific test case generation using higher ordered typed languages for specification*. Maryland, USA: University of Maryland.

STAA, A.v., 2011. *Especificações e teste funcional*. Rio de Janeiro: Departamento de Informática, PUC-Rio.

STAA, A.v., 2013a. *Especificações: resumo*. Rio de Janeiro: PUC-Rio. Disponível em: <http://www.inf.puc-rio.br/~inf1413/>.

STAA, A.v., 2013b. *Máquinas de estado*. Rio de Janeiro: PUC-Rio. Disponível em: <http://www.inf.puc-rio.br/~inf1413/>.

SUTHERLAND, J. & SCHWABER, K., 1995. Business object design and implementation. *OOPSLA*.

TARJAN, R.E., 1976. Edge-disjoint spanning trees and depth-first search. *Acta Informatica* 6, pp.171–85.

WORLD WIDE WEB CONSORTIUM, 1998. *Extensible Markup Language (XML)*. [Online] Available at: <http://www.w3.org/TR/REC-xml/>.

XIE, T. et al., 2009. *Reggae - Automated Test Generation for Programs using Complex Regular Expressions*. Raleigh, USA / Seattle, USA: North Carolina State University / Microsoft Research.

YUAN, X., 2008. *Feedback-directed model-based GUI test case generation*. Maryland, USA: University of Maryland.

YUAN, X. & MEMON, A.M., 2010. Generating Event Sequence-Based Test Cases Using GUI Runtime State Feedback. *IEEE Transactions on Software Engineering*, vol. 36, no. 1, February.

Apêndice A

A listagem a seguir apresenta o conteúdo do arquivo de configuração de dependências do projeto FunTester, `pom.xml`, utilizado pelo Apache Maven. O arquivo contém configurações, repositórios e as respectivas versões utilizadas.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>funtester</groupId>
  <artifactId>funtester</artifactId>
  <packaging>jar</packaging>
  <name>FunTester</name>
  <version>0.7</version>
  <description>Functional Tester</description>

  <properties>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
  </properties>

  <developers>
    <developer>
      <name>Thiago Delgado Pinto</name>
      <email>tpinto@inf.puc-rio.br</email>
    </developer>
  </developers>

  <repositories>

    <repository>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
      <id>mylocalrepo</id>
      <name>Local Repository</name>
      <url>file://C:/dev/m2/</url>
    </repository>

    <repository>
      <id>mvnrepository</id>
      <name>MVN Repository</name>
      <url>mvnrepository.com/artifact</url>
    </repository>

    <repository>
      <id>sonatype-nexus-snapshots</id>
      <name>Sonatype</name>
      <url>http://repository.sonatype.org</url>
    </repository>

    <repository>
      <id>java.net-Public</id>
```

```

        <name>Maven Java Net Snapshots and Releases</name>

        <url>https://maven.java.net/content/groups/public/</url>
    </repository>

    <!-- Repository for the Java Simple Framework Plugin
(JSFP) -->
    <repository>
        <id>maven.formreturn.com</id>
        <name>Form Return</name>
        <url>http://maven.formreturn.com/repository/</url>
    </repository>

</repositories>

<dependencies>

    <!-- Apache Log4J -->
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.17</version>
    </dependency>

    <!-- Apache Commons Lang -->
    <dependency>
        <groupId>commons-lang</groupId>
        <artifactId>commons-lang</artifactId>
        <version>2.6</version>
    </dependency>

    <!-- Apache Commons Exec -->
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-exec</artifactId>
        <version>1.1</version>
    </dependency>

    <!-- SLF4j -->
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.6.4</version>
    </dependency>

    <!-- SLF4j for Log4J -->
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>1.6.4</version>
    </dependency>

    <!-- Automaton (for Xeger) -->
    <dependency>
        <groupId>dk.brics</groupId>
        <artifactId>automaton</artifactId>
        <version>1.11</version>
    </dependency>

    <!-- Xeger -->
    <dependency>
        <groupId>nl.flotsam</groupId>
        <artifactId>xeger</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>

    <!-- Joda-Time -->

```

```

<dependency>
  <groupId>joda-time</groupId>
  <artifactId>joda-time</artifactId>
  <version>2.2</version>
</dependency>

<!-- JGraphT -->
<dependency>
  <groupId>jgrapht</groupId>
  <artifactId>jgrapht</artifactId>
  <version>0.7.3</version>
</dependency>

<!-- Java Simple Framework Plugin (JSFP) -->
<dependency>
  <groupId>net.xeoh.jspf</groupId>
  <artifactId>jspf-core</artifactId>
  <version>1.0.2</version>
</dependency>

<!-- TestNG (unit testing framework) -->
<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>6.8</version>
  <scope>test</scope>
</dependency>

<!-- FEST Assert (matcher library) -->
<dependency>
  <groupId>org.easytesting</groupId>
  <artifactId>fest-assert</artifactId>
  <version>1.4</version>
</dependency>

<!-- Jackson (JSON) -->
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.1.3</version>
</dependency>

<!-- Jackson JODA Time -->
<dependency>
  <groupId>com.fasterxml.jackson.datatype</groupId>
  <artifactId>jackson-datatype-joda</artifactId>
  <version>2.1.2</version>
</dependency>

<!-- JGoodies -->
<dependency>
  <groupId>com.jgoodies</groupId>
  <artifactId>forms</artifactId>
  <version>1.2.1</version>
</dependency>

<!-- Substance Swing Look and Feel -->
<dependency>
  <groupId>org.java.net.substance</groupId>
  <artifactId>substance</artifactId>
  <version>6.0</version>
</dependency>

<!-- JFreeChart -->
<dependency>
  <groupId>org.jfree</groupId>
  <artifactId>jfreechart</artifactId>
  <version>1.0.14</version>

```

```

        </dependency>

        <!-- JDBC CONNECTORS -->

        <!-- MySQL JDBC -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.6</version>
        </dependency>

        <!-- Firebird JDBC (Jaybird) -->
        <dependency>
            <groupId>org.firebirdsql.jdbc</groupId>
            <artifactId>jaybird</artifactId>
            <version>2.1.6</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <!-- Maven JAR Plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>2.2</version>
                <!-- nothing here -->
            </plugin>

            <!-- Maven Assembly Plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-assembly-
plugin</artifactId>
                <version>2.2-beta-4</version>
                <configuration>
                    <descriptorRefs>
                        <descriptorRef>jar-with-
dependencies</descriptorRef>
                    </descriptorRefs>
                    <archive>
                        <manifest>

                            <mainClass>Main</mainClass>

                        </manifest>
                    </archive>
                </configuration>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>

            <!-- Maven Compiler Plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-
plugin</artifactId>
                <configuration>
                    <source>1.6</source>
                    <target>1.6</target>

```

```
                </configuration>
            </plugin>
        </plugins>
    </build>

    <reporting>
        <plugins>
            <!-- Cobertura Maven Plugin -->
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>cobertura-maven-
plugin</artifactId>
                <version>2.5.1</version>
            </plugin>
        </plugins>
    </reporting>
</project>
```

Anexo A

O conteúdo a seguir apresenta uma listagem parcial da documentação de especificação *original* do software PetShop Admin, usada na avaliação da ferramenta. Nela, o caso de uso "Efetuar Login" é chamado de "Login de Usuário" e não há a descrição do caso de uso "Remover Animal". Os casos de uso não utilizados na avaliação foram omitidos.

IMPORTANTE: A descrição dos casos de uso abaixo é apresentada tal qual como foi encontrada na Internet, isto é, *sem quaisquer correções*.

DESCRIÇÃO GERAL DO SISTEMA

O sistema visa facilitar o controle da entrada e saída de produtos e o lucro obtido por produtos e serviços.

Fácil de ser utilizado mesmo para pessoas que não entendem muito de informática, pois conta com uma interface intuitiva, o que facilita sua utilização por qualquer usuário.

Além disso, informações sobre produtos comprados e serviços prestados ficam gravadas no cadastro do cliente, o que facilita e melhora o atendimento e as tentativas de fidelização do mesmo.

1. REQUISITOS FUNCIONAIS

[RF01] Login de Usuário

Prioridade: Essencial

Descrição do requisito funcional: O sistema deve apresentar ao usuário uma tela de "Login" para permitir acesso à janela principal do sistema. A autenticação do login só deve ocorrer para usuários já cadastrados.

Pré-condição: O funcionário inserir nome e senha corretos.

Pós-condição: A janela principal será exibida para o usuário. Os outros requisitos poderão ser executados.

Usuário: Funcionário e Gerente

Fluxo principal do evento:

1. O funcionário informa os dados necessários para o login.
 - Nome de usuário;
 - Senha do usuário;
2. O sistema verifica se os dados do login são válidos.
3. O sistema carrega as informações do usuário. (funções disponíveis ao usuário)
4. O sistema configura a janela principal de acordo com as informações do usuário.
5. A janela de login é fechada.
6. A janela principal é exibida com as funções disponíveis.

Fluxo secundário:

1. No passo 2 do fluxo principal, se a verificação for inválida, o sistema deve apresentar uma caixa de diálogo com a mensagem "Usuário ou senha inválidos!" e retorna para o passo 1 do fluxo principal.

[RF02] Cadastrar Cliente

Prioridade: Essencial

Descrição do requisito funcional: O sistema deve apresentar uma janela onde o usuário entrará com as informações do cliente.

Pré-condição: O usuário ter permissão para cadastrar clientes.

Pós-condição: Os dados do cliente são validados e o cliente é inserido no sistema.

Usuário: Funcionário

Fluxo principal do evento:

1. O funcionário informa os dados do cliente a ser cadastrado.
 - CPF;
 - RG;
 - Nome;

- *Sexo;*
 - *Telefone;*
 - *Celular;*
 - *Rua;*
 - *Nº da residência;*
 - *Bairro;*
 - *Cidade;*
 - *UF;*
 - *CEP;*
 - *Complemento;*
 - *Descrição;*
2. *O sistema verifica as informações digitadas.*
 3. *O sistema inicia o [RF03].*
 4. *O cliente é inserido no banco de dados do sistema.*

Fluxos secundários:

1. *No passo 2 do fluxo principal, ocorre uma validação do CPF (informa se o CPF é válido ou não). Caso seja inválido, o sistema mostrará a mensagem “CPF inválido!” e retornará ao passo 1 do fluxo principal.*
2. *No passo 3 do fluxo principal, se o cliente existir no banco de dados, ou seja, já for cadastrado, o sistema mostrará a mensagem “Cliente já cadastrado!” e retornará ao passo 1 do fluxo principal.*
3. *Em qualquer momento a operação pode ser cancelada.*

[RF05] Cadastrar Animal

Prioridade: Essencial

Descrição do requisito funcional: O sistema deve apresentar uma janela onde o usuário entrará com as informações do animal de estimação que será cadastrado.

Pré-condição: O animal só poderá ser cadastrado se tiver um cliente cadastrado.

Pós-condição: O animal é inserido no sistema.

Usuário: Funcionário

Fluxo principal do evento:

1. *O usuário seleciona um cliente.*
2. *O usuário informa os dados do animal a ser cadastrados.*

- *Registro do dono;*
 - *Nome do animal;*
 - *Tipo;*
 - *Raça;*
 - *Idade;*
 - *Sexo;*
3. *O sistema verifica as informações digitadas.*
 4. *O animal cadastrado*

Fluxos secundários

1. *No passo 3, se alguma informação for inválida, o sistema exibirá a mensagem “Dados inválidos”, e retornará para o passo 2.*

Em qualquer momento a operação pode ser cancelada.
