UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

SÉRGIO LUIS BARBIERI

Teste de software na indústria: Um estudo qualitativo

São Paulo

2018

SÉRGIO LUIS BARBIERI

Teste de software na indústria: Um estudo qualitativo

Versão original

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Área de concentração: Metodologia e Técnicas da Computação

Orientador: Prof. Dr. Marcos Lordello Chaim

São Paulo

2018

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

CATALOGAÇÃO-NA-PUBLICAÇÃO

(Universidade de São Paulo. Escola de Artes, Ciências e Humanidades. Biblioteca)
CRB 8 - 4936

Barbieri, Sérgio Luis

Teste de software na indústria : um estudo qualitativo / Sérgio Luis Barbieri ; orientador, Marcos Lordello Chaim. – 2018.
76 f. : il

Dissertação (Mestrado em Ciências) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo.

Versão original

1. Desenvolvimento de software. 2. Teste e avaliação de software. I. Chaim, Marcos Lordello, orient. II. Título

CDD 22.ed.– 005.12

Dissertação de autoria de Sérgio Luis Barbieri, sob o título “**Teste de software na indústria: Um estudo qualitativo**”, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação, na área de concentração Metodologia e Técnicas da Computação, aprovada em _____ de _____ de _____ pela comissão julgadora constituída pelos doutores:

_____
Prof. Dr. _____

Instituição: _____

Presidente

_____
Prof. Dr. _____

Instituição: _____

_____
Prof. Dr. _____

Instituição: _____

_____
Prof. Dr. _____

Instituição: _____

*Dedico o meu trabalho de conclusão para a minha família que me apoiou nessa nova
jornada.*

Agradecimentos

Gostaria de agradecer primeiramente ao meu orientador, Prof. Dr. Marcos Lordello Chaim, por ter abraçado o meu projeto desde o início e me auxiliado no desenvolvimento do mesmo. Quero também agradecer aos meus colegas de sala de aula, de profissão e demais professores que muito contribuíram na minha formação. E, por fim, um agradecimento especial à minha filha Marcella Barbieri que me ajudou quando precisei.

Resumo

BARBIERI, Sérgio Luis. **Teste de software na indústria:** Um estudo qualitativo. 2018. 76 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2018.

Em projetos de desenvolvimento a qualidade do produto de software é fundamental para o seu sucesso. Uma das etapas que busca garantir a qualidade do produto final é a validação, verificação e teste (VV&T). O teste é uma das atividades chave de VV&T realizada para garantir a qualidade. Técnicas de teste foram desenvolvidas para verificar e validar tanto requisitos funcionais como não funcionais. Além disso, essas técnicas são aplicadas nas organizações por meio de estratégias que definem os tipos de teste que serão realizados, a ordem de realização, a sua automatização e a frequência de ocorrência. Uma questão importante é verificar como essa atividade é realizada na indústria de software, como essas técnicas são utilizadas e a sua adoção em organizações desenvolvedoras de software. Este trabalho apresenta uma pesquisa qualitativa da atividade de teste. Ela utilizou a técnica teoria fundamentada aplicada em cinco empresas desenvolvedoras de software para estabelecer uma teoria da atividade de teste. Foram realizadas entrevistas com gestores de teste das empresas e, a partir dessas entrevistas, foi desenvolvido uma teoria sobre a organização da atividade de teste. Nesta teoria, observou-se que a estrutura organizacional direciona a escolha das técnicas e ferramentas, bem como o tipo do sistema; o prazo e orçamento condicionam a utilização de técnicas de automatização.

Palavras-chaves: Teste de software. Atividade de teste. Teoria fundamentada. Estudo qualitativo. Estudo de caso na indústria.

Abstract

BARBIERI, Sérgio Luis. **Software testing at industrial settings:** A qualitative study. 2018. 76 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2018.

In development projects the quality of the software product is critical to its success. One of the steps that enforces the quality of the final product is validation, verification and testing (VV&T). Testing is one of the key VV&T activities to ensure quality. Testing techniques were developed to verify and validate both functional and non-functional requirements. In addition, these techniques are applied in organizations using strategies that define the testing techniques that will be performed, the order of application, their automation and the frequency of occurrence. An important issue is to verify how testing is carried out in the software industry, how these techniques are used and their adoption in industrial settings. This work presents a qualitative research of the testing activity. It used the grounded theory technique applied in five software development organizations to establish a model of the testing activity. Interviews were conducted with testing managers and, from these interviews, an organizational theory of the testing activity was developed. In this theory, it was observed that the organizational structure directs the choice of techniques and tools, as well as the type of the system; project schedule and budget limit the use of automation techniques.

Keywords: Software testing. Testing activity. Grounded theory. Qualitative study. Industrial case study.

Lista de figuras

Figura 1 – Programa exemplo <code>calcularMDC</code>	19
Figura 2 – Mapeamento de linhas para nós do programa exemplo <code>calcularMDC</code> . .	27
Figura 3 – Grafo de fluxo de controle do método <code>calcularMDC()</code>	28
Figura 4 – Processo de pesquisa teoria fundamentada	49
Figura 5 – Teoria resultante da análise dos dados	55
Figura 6 – Modelo organizacional	57
Figura 7 – Modelo e estrutura do projeto	58
Figura 8 – Abordagem de teste	59
Figura 9 – Classe de teste do método <code>calcularMDC</code>	68
Figura 10 – Cobertura de complexidade utilizando a EclEmma/JaCoCo	69
Figura 11 – Cobertura de instruções utilizando a EclEmma/JaCoCo	70
Figura 12 – Cobertura de ramos utilizando EclEmma/JaCoCo	70
Figura 13 – Resumo da análise de cobertura realizada pela EclEmma/JaCoCo . . .	71
Figura 14 – Análise de cobertura realizada pela EclEmma/JaCoCo no código fonte	72
Figura 15 – Trecho de código para cobertura de 100% do método <code>calcularMDC</code> . .	72

Lista de quadros

Quadro 1 – Trabalhos relacionados	34
Quadro 2 – Guia Entrevista	50
Quadro 3 – Empresas participantes x segmento	53
Quadro 4 – Profissional x Experiência	53
Quadro 5 – Conceitos e categorias	54

Lista de tabelas

Tabela 1 – Conjunto de casos de teste	21
Tabela 2 – Possível conjunto de requisitos de teste estabelecidos pelo critério baseado em complexidade para o programa exemplo.	29
Tabela 3 – Requisitos de teste estabelecidos pelos critérios todos os nós e todos os ramos para o programa exemplo.	29

Lista de abreviaturas e siglas

IDE	Integrated Development Environment
PLUGIN	Componente externo incorporado a um programa
IEEE	Institute of Electrical and Electronics Engineers
API	Application Programming Interface
TDD	Test-driven development

Sumário

1	Introdução	15
1.1	<i>Motivação</i>	16
1.2	<i>Justificativa</i>	16
1.3	<i>Objetivos</i>	16
1.4	<i>Contribuições</i>	17
1.5	<i>Estrutura desse projeto</i>	17
2	Conceitos	18
2.1	<i>Engano, defeito, erro, falha e caso de teste</i>	18
2.1.1	Engano	18
2.1.2	Defeito	18
2.1.3	Erro ou infecção	20
2.1.4	Falha	20
2.1.5	Caso de teste	20
2.2	<i>Teste de software no desenvolvimento de software</i>	21
2.3	<i>Estratégias de testes de software</i>	23
2.3.1	Testes de unidade	23
2.3.2	Testes de integração	23
2.3.3	Testes de regressão	24
2.3.4	Testes de aceitação	24
2.4	<i>Técnicas de requisitos funcionais de software</i>	24
2.4.1	Técnicas caixa preta	24
2.4.2	Técnicas caixa branca	25
2.4.3	Análise de fluxo de controle	26
2.4.4	Conceitos de fluxo de controle	26
2.4.5	Critérios Baseados em Fluxo de Controle	26
2.5	<i>Teste de requisitos não funcionais</i>	29
2.5.1	Teste de desempenho	30
2.5.2	Teste de usabilidade	30
2.5.3	Teste de segurança	30

2.6	<i>Automatização do teste de software</i>	31
2.7	<i>Considerações finais</i>	32
3	Trabalhos relacionados	33
3.1	<i>Teste de software e teoria fundamentada</i>	33
3.2	<i>Seleção dos trabalhos</i>	33
3.3	<i>Descrição dos trabalhos selecionados</i>	34
3.3.1	Teste durante e ao final do desenvolvimento	34
3.3.2	Atividades de teste em empresas	35
3.3.3	Atividades de teste e métodos ágeis	37
3.3.4	Teste de aplicativos plugin	37
3.3.5	Melhoria da qualidade de software	38
3.3.6	Uso de teoria fundamentada em teste de software	39
3.3.7	Complexidade do teste	40
3.3.8	Testes de segurança	40
3.4	<i>Considerações finais</i>	41
4	Procedimentos metodológicos	42
4.1	<i>Coleta de dados</i>	43
4.2	<i>Codificação</i>	45
4.2.1	Codificação aberta	46
4.2.2	Codificação axial	47
4.2.3	Codificação seletiva	48
4.3	<i>Projeto</i>	48
4.3.1	Projeto para coleta de dados	49
4.3.2	Análise de dados	50
4.4	<i>Considerações finais</i>	50
5	Resultados do estudo qualitativo	52
5.1	<i>Descrição do processo de seleção das empresas e dos indivíduos entre-</i> <i>vistados</i>	52
5.2	<i>Descrição do processo de coleta de dados</i>	53
5.3	<i>Descrição do processo de análise de conteúdo</i>	53
5.4	<i>Resultados da análise de conteúdo</i>	54

5.4.1	Modelo organizacional	56
5.4.2	Modelo e estrutura do projeto	56
5.4.3	Abordagem de teste	58
5.5	<i>Teoria resultado da aplicação da análise fundamentada</i>	61
5.6	<i>Ameaças à validade</i>	61
5.7	<i>Considerações finais</i>	62
6	Conclusões	63
6.1	<i>Síntese do trabalho</i>	63
6.2	<i>Contribuições</i>	64
6.3	<i>Trabalhos futuros</i>	64
	Referências¹	65
	Apêndice A – Utilização da ferramenta JaCoCo	68
A.1	<i>Relatórios Gerados e Análise dos Dados Produzidos</i>	69
	Apêndice B – Termo de consentimento livre e esclarecido . .	73

---

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

A qualidade no desenvolvimento de software é fundamental para o sucesso de um projeto, tanto na concepção do produto como nas atividades de manutenção e desenvolvimento de novas funcionalidades. A qualidade está fortemente relacionada ao atendimento dos requisitos do projeto por meio de artefatos gerados com exatidão e com confiabilidade (MATHUR, 2009).

Dentro do ciclo de desenvolvimento, a atividade de teste é realizada para garantir o comportamento do software conforme o esperado (MATHUR, 2009). A atividade de teste pode ser dividida a grosso modo em teste de requisitos funcionais e não funcionais. Há diferentes tipos de teste desses requisitos, bem como estratégias para sua realização.

Exemplos de técnicas de teste de requisitos funcionais são as técnicas caixa preta e caixa branca (DELAMARO M. E.; MALDONADO, 2007). O teste caixa preta baseia-se na observação do resultado da saída conforme a entrada de dados; ela é também conhecida como “técnica funcional” porque deriva seus requisitos de teste a partir dos requisitos e funções do sistema. Essa técnica muitas vezes é inviabilizada em função das infinitas possibilidades combinatórias dos dados de entrada (MYERS G. J.; SANDLER, 2004).

A técnica caixa branca, também chamada de “técnica estrutural”, por sua vez, utiliza a implementação para determinar o que deve ser testado. Essa técnica é utilizada de duas maneiras: para criar novos casos de teste e para avaliar um conjunto de casos de teste existente. Para tanto, utiliza critérios baseados em cobertura de código para determinar os requisitos de teste, isto é, o que deve ser testado, e também se esses requisitos de teste são verificados por um conjunto de casos de teste já existente.

Exemplos de requisitos de teste de técnicas estruturais são: executar todas as linhas de um programa ou todas as saídas dos comandos de transferência de fluxo de controle (e.g., if, while, for) ao menos uma vez. Os critérios de teste estruturais têm por objetivo definir quais estruturas devem ser exercitadas pelos casos de teste. Eles são também métricas para avaliar se o teste está adequado (MATHUR, 2009).

As técnicas descritas acima foram criadas para validar os requisitos funcionais de um sistema. Porém, a validação dos requisitos não funcionais é igualmente importante. Eles são também avaliados por meio de teste. Técnicas de teste que avaliam o desempenho, a segurança, a usabilidade e sua adequação do software à carga estabelecida são essenciais

para atender as expectativas dos interessados (cliente e usuários) no sistema. Essas técnicas derivam seus requisitos das restrições do software.

As técnicas de teste de requisitos funcionais e não funcionais são aplicadas no nível de unidade, de módulos ou do sistema como um todo. Elas ainda devem ser organizadas por meio de estratégias de teste que definem a ordem de realização dos testes, a frequência de ocorrência e sua automatização.

1.1 Motivação

Uma questão importante é verificar como a atividade de teste ocorre em ambientes reais de desenvolvimento. Isto porque como elas são aplicadas na indústria podem variar intensamente dependendo de fatores como orçamento de teste, método de desenvolvimento utilizado, perfil do cliente, tipo de software, entre outros.

Questões como quais técnicas, tipos e estratégias de teste são realizadas pela indústria de software, como os testes são organizados e o que determina essa organização são relevantes para determinar como eles ocorrem na prática. A partir desse conhecimento, pode-se traçar estratégias para torná-los mais eficazes e eficientes. Por isso, é importante avaliar como a atividade de teste ocorre, em especial, no contexto de empresas brasileiras desenvolvedoras de software.

1.2 Justificativa

Várias técnicas e estratégias de teste foram propostas na literatura. No entanto, poucos são os estudos que avaliam como elas são realizadas na prática. O entendimento de como a atividade de teste dá-se em ambientes industriais de desenvolvimento é fundamental para avaliar sua efetividade e para o desenvolvimento de novas técnicas e estratégias mais efetivas.

1.3 Objetivos

O objetivo desta pesquisa é verificar como a atividade de teste é realizada em ambientes industriais de desenvolvimento. Ao final, pretende-se construir um modelo de como os testes ocorrem na indústria.

Para elaborar esse modelo, foi utilizada metodologia de pesquisa qualitativa (STRAUSS A.;CORBIN, 2008). Os objetivos específicos deste trabalho são os seguintes:

- identificar organizações desenvolvedoras de software com atividades de teste organizadas;
- identificar os responsáveis pela atividade de teste;
- coletar os dados por meio de entrevistas;
- elaborar o modelo da atividade de teste utilizando metodologia qualitativa de pesquisa, em particular, a técnica conhecida como teoria fundamentada (STRAUSS A.;CORBIN, 2008).

1.4 Contribuições

Este trabalho entrevistou gestores de cinco empresas e, a partir dessas entrevistas, foi desenvolvido uma teoria da atividade de teste em empresas brasileiras. Neste modelo, observou-se que a estrutura organizacional direciona a escolha das técnicas e ferramentas, bem como o tipo do sistema e a estrutura do projeto; o prazo e orçamento condicionam a utilização de técnicas de automatização.

1.5 Estrutura desse projeto

Este projeto é organizado da seguinte forma. Nessa seção, foram apresentados a motivação, a justificativa, os objetivos da pesquisa e as contribuições esperadas. No próximo capítulo, serão apresentados os conceitos básicos; o Capítulo 3 contém uma revisão de literatura sobre os trabalhos relacionais. O Capítulo 4 apresenta os procedimentos metodológicos e, no Capítulo 5, são apresentados os resultado experimentais. O Capítulo 6 apresenta as conclusões e trabalhos futuros.

2 Conceitos

O objetivo do teste é provocar falhas que revelam a presença de defeitos. Neste capítulo, são descritos sucintamente os conceitos e técnicas de teste de software utilizados pelos desenvolvedores. Primeiramente, definem-se os conceitos de engano, defeito, erro, falha e caso de teste. Em seguida, são apresentados aqueles associados aos diferentes tipos, estratégias e técnicas de teste.

2.1 Engano, defeito, erro, falha e caso de teste

Os termos engano, defeito, erro ou infecção, falha e caso de teste são importantes para a compreensão dos problemas em teste de software. A distinção entre eles é apresentada abaixo com base no Glossário de Termos de Engenharia de Software do IEEE (*Institute of Electrical and Electronic Engineers*) (IEEE, 1990) e nos trabalhos de Ammann e Offutt (2017) e Zeller (2005).

2.1.1 Engano

Engano é uma ação humana que gera um defeito. Essa ação humana pode ocorrer devido a distrações, a erros de digitação e ao cansaço do programador, mas também por falta de conhecimento ou compreensão incorreta dos requisitos, dos possíveis estados do programa e das interfaces dos módulos.

2.1.2 Defeito

O defeito é a manifestação física do engano, ou seja, o trecho incorreto do código do programa. Para ilustrar os conceitos, foi elaborado o programa `calcularMDC`, descrito na Figura 1, que recebe como parâmetro dois números inteiros e calcula o maior divisor comum entre eles. O programa deve retornar o valor encontrado, caso haja um divisor comum, ou retornar -1, caso uma das entradas seja zero.

Um defeito foi inserido no código da linha 14 de forma que, caso o valor inserido no parâmetro *valor1* seja maior que o inserido no parâmetro *valor2*, tanto a variável auxiliar

maiorValor como a variável *menorValor* recebem o conteúdo de *valor2*. Esse defeito é ilustrativo, mas pode ocorrer por falta de atenção do programador, isto é, um engano, ao copiar e colar trechos de código.

O defeito somente será executado quando o valor *valor1* for maior que *valor2*, caso contrário, este defeito não ocasiona uma saída incorreta.

Figura 1 – Programa exemplo `calcularMDC`

```

1 package Programa.Exemplo;
2
3 public class MDC {
4
5     public static int calcularMDC(int valor1, int valor2) {
6         int ret = -1;
7         int maiorValor;
8         int menorValor;
9         int resto;
10
11         if (valor1 != 0 && valor2 != 0) {
12             if (valor1 != valor2) {
13                 if (valor1 > valor2) {
14                     maiorValor = valor2; // <- defeito
15                     menorValor = valor2;
16                 } else {
17                     maiorValor = valor2;
18                     menorValor = valor1;
19                 }
20                 resto = maiorValor mod menorValor;
21                 while (resto != 0) {
22                     maiorValor = menorValor;
23                     menorValor = resto;
24                     resto = maiorValor mod menorValor;
25                 }
26                 ret = menorValor;
27             } else
28                 ret = valor1;
29         }
30     }
31     return ret;
32 }
33 }
```

2.1.3 Erro ou infecção

Um erro ou infecção ocorre quando um estado de programa não previsto é atingido quando o defeito é executado sob certas condições. Um erro ou infecção pode causar mais erros ou infecções em partes de código sem defeitos.

No programa `calculaMDC` da Figura 1, o erro é disparado quando a linha 14 é executada, ou seja, quando a condição $valor1 > valor2$ for verdadeira e a variável auxiliar *maiorValor* receber *valor2* ao invés *valor1*. Dessa forma, o programa apresenta um estado que não é o correto.

Uma vez que existe um erro, uma falha pode ocorrer. Porém, assim como o defeito, um erro pode existir, mas não há garantias de ser observada a falha.

2.1.4 Falha

A falha é um erro ou infecção observável externamente. Um erro propaga-se e gera um comportamento anormal do sistema. Uma falha é visível aos usuários finais como uma mensagem de erro ou saídas incorretas.

No programa `calculaMDC`, a falha ocorre quando o valor do primeiro parâmetro for maior que o segundo parâmetro. Como na linha 14 e na linha 15, o mesmo valor é atribuído às variáveis auxiliares *maiorValor* e *menorValor*, o máximo divisor comum será o *valor2*, ocasionando uma saída incorreta do programa.

De acordo com Dijkstra, os testes podem apenas mostrar a presença de defeitos, mas nunca sua ausência (apud Zeller (2005)). Se defeitos não causam erro (infecção) que geram falhas, todos os casos de teste irão passar.

2.1.5 Caso de teste

As falhas somente ocorrem quando o software em teste é executado. Quando o programa falha, sabe-se que o defeito foi atingido, provocou um erro que se manifestou por meio da falha observada. O programa é executado por casos de teste.

Caso de teste, ou simplesmente teste, é um artefato composto de vários componentes.

Ammann e Offutt (2017) definiram formalmente a composição de um caso de teste. Um caso de teste é composto por valores entrada, valores pré-execução (*prefix values*), valores pós-execução (*posfix values*) e resultados esperados. Esses componentes são necessários para execução e avaliação do software em teste. Os valores pré-execução são necessários para colocar o software em condições de receber os valores de entrada. Os valores pós-execução são necessários para verificar os resultados e para terminar o programa ou colocá-lo em um estado estável.

Tabela 1 descreve casos de teste para o programa da Figura 1. Nota-se que o programa `calculaMDC` é simples e, por isso, seus casos de teste requerem apenas os valores de entrada e as saídas esperadas. A tabela apresenta também o resultado dos testes: sucesso ou falha.

Um conjunto de teste, ou uma suíte de teste, é definido por um conjunto de casos de teste. Um *script* de teste é um programa para executar automaticamente um caso de teste e produzir um relatório.

Tabela 1 – Conjunto de casos de teste

T_n	Caso de teste	Resultado esperado	Resultado obtido	Sucesso/Falha
t_1	<code>calculaMDC(8,16)</code>	8	8	✓
t_2	<code>calculaMDC(7,4)</code>	1	4	✗
t_3	<code>calculaMDC(10,10)</code>	10	10	✓
t_4	<code>calculaMDC(0,2)</code>	-1	-1	✓
t_5	<code>calculaMDC(2,0)</code>	-1	-1	✓

Fonte: Vincenzi et al. (2018)

2.2 Teste de software no desenvolvimento de software

Os processos de desenvolvimento de software incluem atividades comuns como: especificação de requisitos, desenvolvimento do sistema, validação e verificação (V&V) e evolução do software (SOMMERVILE, 2011). Geralmente, estas atividades são implementadas seguindo a lógica de um modelo de desenvolvimento de software. De acordo com Sommerville (2011), vários modelos de processos de desenvolvimento são utilizados no desenvolvimento de sistemas de software. A seguir, apresentam-se alguns deles.

- Modelo em cascata: as atividades são divididas em etapas e implementadas sequencialmente.

- Modelo iterativo e incremental: o desenvolvimento é dividido em passos similares; a cada etapa do processo são acrescentadas novas funcionalidades.
- Modelo formal: as atividades são modeladas e implementadas seguindo um modelo matemático formal.
- Modelo baseado em componentes: baseada em reúso de aplicações e artefatos de software já desenvolvidos.

O teste de software está inserido dentro das atividades de verificação e validação (AMMANN; OFFUTT, 2017). Verificação é o processo de determinar se o produto de uma fase de desenvolvimento preenche os requisitos estabelecidos na fase anterior. Validação, por sua vez, avalia se o produto gerado no final do processo de desenvolvimento está de acordo com o uso pretendido. As atividades de teste têm por objetivo verificar e validar os artefatos gerados durante o desenvolvimento de software.

A importância do teste de software aumentou consideravelmente com o surgimento dos métodos ágeis (BECK; ANDRES, 2004). O problema fundamental dos modelos de desenvolvimento tradicionais, baseados em planificação e documentação extensa, é que eles não conseguem lidar adequadamente com as mudanças no contexto no desenvolvimento de software. É comum durante o desenvolvimento as condições de mercado mudarem, as necessidades dos usuários não serem mais as mesmas, a equipe sofrer alterações e novas ameaças competitivas surgirem (PRESSMAN, 2006). Os métodos ágeis têm se tornado prevalentes na indústria de software porque utilizam práticas baseadas em pouca documentação, interação com o cliente, respostas rápidas a mudanças, muita comunicação verbal e foco na geração de código executável da aplicação.

No contexto ágil, os testes¹ ocorrem cedo e frequentemente no processo de desenvolvimento (MCGREGOR, 2007). Beck (2002) preconizam que todo requisito do software deve ser acompanhado de um caso de teste automatizado. Mais ainda que os testes devem ser automatizados para que produzam *feedback* constante do estado do software em desenvolvimento. Por isso, práticas de desenvolvimento dirigido a testes (TDD – *Test-driven development*) e de automatização da execução dos casos de teste são fundamentais para o sucesso dos métodos ágeis (BECK, 2002).

---

¹ O termo *testes*, no plural, é usado de uma maneira ampla neste documento. O objetivo foi deixar o texto mais fluente, pois acredita-se que o leitor é capaz de fazer a distinção a partir do contexto.

2.3 Estratégias de testes de software

A aplicação de testes corresponde à principal técnica de Verificação e Validação (V&V) de software (SOMMERVILE, 2011). Verificação corresponde, entre outras atividades, ao processo de teste das funcionalidades implementadas, enquanto o processo de validação analisa se o software desenvolvido atende aos requisitos do cliente.

As subseções seguintes descrevem algumas estratégias de testes aplicadas durante o processo de V&V (SOMMERVILE, 2011):

2.3.1 Testes de unidade

Testes de unidade, também conhecidos como testes unitários, são verificações aplicadas a componentes ou módulos desenvolvidos. O termo unitário é utilizado, pois componentes e módulos (conhecidos como unidades) são testados individualmente. Sendo aplicados a componentes internos do sistema, estes testes são geralmente realizados por desenvolvedores, podendo ser utilizadas técnicas caixa preta ou caixa branca.

2.3.2 Testes de integração

Após a aplicação de testes de unidade, há a necessidade de testar o funcionamento do sistema quando diferentes módulos do sistema são integrados. Mesmo funcionando corretamente em separado, os módulos podem apresentar problemas quando integrados devido a dependências existentes no fluxo de dados, nas chamadas e retornos de funções e nas interfaces existentes. Neste tipo de teste, podem ser aplicadas técnicas caixa preta, branca ou cinza.

O teste de integração tem como foco a “interface” entre os componentes, garantindo o comportamento correto quando executados em conjunto. Diferente do teste de unidade, na integração o resultado gerado por um ou mais componente pode estar em um estado não aceitável. O objetivo desta etapa é garantir que o conjunto de componentes funcionem conforme desejado em conjunto (MATHUR, 2009).

2.3.3 Testes de regressão

Outra abordagem muito utilizada são os chamados testes de regressão, a cada integração realizada, testes em funcionalidades já implementadas e testadas são reexecutados. A intenção deste tipo de teste é verificar se os novos módulos implementados não danificaram o funcionamento correto de componentes que já haviam sido validados. Devido à característica repetitiva dos testes de regressão, é essencial que estes sejam automatizados.

2.3.4 Testes de aceitação

Estes testes têm a finalidade de mostrar que o software corresponde ao que o cliente deseja, ou seja, que o software atende a seus requisitos. Eles podem ser realizados pela equipe do cliente ou por uma organização contratada para realizar os testes de aceitação.

2.4 *Técnicas de requisitos funcionais de software*

A principal meta de qualquer técnica de teste de software é executar o programa na intenção de provocar falhas (PRESSMAN, 2006). Esta subseção apresenta as principais técnicas voltadas para o teste dos requisitos funcionais do software.

2.4.1 Técnicas caixa preta

A técnica caixa preta ou funcional projeta casos de testes de forma que o programa é considerado uma caixa preta e a geração dos casos de testes dá-se a partir dos requisitos do sistema. Nesta técnica, não se consideram os detalhes da construção do software (DELAMARO M. E.; MALDONADO, 2007).

O teste funcional utiliza os requisitos do sistema para a criação dos casos de teste. A saída obtida após a entrada de dados é comparada com a saída esperada de acordo com os requisitos. Nessa técnica, a implementação do código não é avaliada. A técnica funcional muitas vezes é inviabilizada em função das infinitas possibilidades combinatórias dos dados de entrada (MYERS G. J.; SANDLER, 2004).

Considerando a impossibilidade de submeter um programa a todas as possíveis entradas, existem critérios para estabelecer entradas adequadas para teste. Os critérios mais conhecidos são: particionamento de equivalência e análise do valor limite.

A seguir, descrevem essas duas técnicas.

Particionamento de equivalência

A partir das definições do sistema e requisitos, o objetivo é reduzir as entradas para teste por meio da busca de classes de equivalência: dados que são tratadas da mesma maneira resultando em um comportamento similar, reduzindo assim o número de entradas. A intuição é que um dado de entrada em uma classe equivalente terá o mesmo tratamento (sucesso ou falha) em relação a outros que pertencem a mesma classe.

Análise do valor limite

A experiência mostra que casos de teste que exploram condições limites têm maior probabilidade de encontrar defeitos (MYERS G. J.;SANDLER, 2004). Esse critério pode ser combinado com o particionamento de equivalência, utilizando valores que dividem classes de equivalência ou valores imediatamente acima ou abaixo.

2.4.2 Técnicas caixa branca

A técnica caixa branca ou estrutural estabelece os requisitos de teste com base em uma determinada implementação (MYERS G. J.;SANDLER, 2004). O teste estrutural tem como objetivo orientar na construção de casos de testes, baseados em caminhos lógicos (decisões e laços) e pares de definições e uso de variáveis, sempre com o foco em revelar novos defeitos.

São apresentadas a seguir técnicas de teste caixa branca baseados em análise de fluxo de controle, pois são as mais utilizadas na indústria.

2.4.3 Análise de fluxo de controle

A técnica de teste estrutural requer que os casos de teste selecionados executem (exercitem) determinados caminhos do programa considerados relevantes para que o testador aumente a sua confiança em relação à correção do programa.

Os caminhos requeridos definem requisitos de teste que constituem um critério de adequação do conjunto de casos de teste. Em outras palavras, o conjunto de casos de teste T é considerado adequado, do ponto de vista de teste estrutural, de acordo com um critério de teste C , se o conjunto de requisitos de teste (TR_C) estabelecido por C é exercitado pelos casos de teste de T .

Os conceitos de fluxo de controle utilizados para definir os requisitos de teste, bem como principais critérios de teste de fluxo de controle são descritos a seguir.

2.4.4 Conceitos de fluxo de controle

Um programa P escrito em uma linguagem procedimental pode ser mapeado em um grafo de fluxo de controle $G(N, B, s, e)$ onde N é o conjunto de blocos de comandos (nós) que são executados sequencialmente, e é o nó de entrada, s é o nó de saída e R é o conjunto de ramos que representam a possível transferência de fluxo de controle entre dois nós.

A Figura 3 apresenta o grafo de fluxo de controle obtido a partir do programa `calcularMDC` (Figura 1). O mapeamento das linhas do programa e os respectivos nós é apresentando na forma de comentários `/* Nó */` no Programa 2. Assim, as linhas 5 a 9 do programa são mapeadas para o nó 1 como mostra o trecho de código. A possível transferência de fluxo de controle do nó 2 (condição do primeiro `if`) para o nó 3 (condição do segundo `if`) é representada pelo ramo (2,3) na Figura 3.

2.4.5 Critérios Baseados em Fluxo de Controle

Os requisitos de teste desses critérios estão associados aos nós, ramos e caminhos do grafo de fluxo de controle do programa.

Figura 2 – Mapeamento de linhas para nós do programa exemplo calcularMDC

```

1 package Livro.Exemplo;                                /* No */
2
3 public class MDC {
4
5     public static int calcularMDC(int valor1, int valor2){/* 1 */
6         int ret = -1;                                   /* 1 */
7         int maiorValor;                                 /* 1 */
8         int menorValor;                                /* 1 */
9         int resto;                                     /* 1 */
10
11         if (valor1 != 0 && valor2 != 0) {                /* 2 */
12             if (valor1 != valor2) {                     /* 3 */
13
14                 if (valor1 > valor2) {                   /* 4 */
15                     maiorValor = valor1;                 /* 5 */
16                     menorValor = valor2;                 /* 5 */
17                 } else {
18                     maiorValor = valor2;                 /* 6 */
19                     menorValor = valor1;                 /* 6 */
20                 }
21
22                 resto = maiorValor mod menorValor;       /* 7 */
23                 while (resto != 0) {                     /* 8 */
24                     maiorValor = menorValor;             /* 9 */
25                     menorValor = resto;                  /* 9 */
26                     resto = maiorValor mod menorValor;   /* 9 */
27                 }
28                 ret = menorValor;                         /* 10 */
29             } else
30                 ret = valor1;                             /* 11 */
31         }
32     }
33
34     return ret;                                         /* 12 */
35 }
36 }

```

Fonte: Sérgio Luis Barbieri, 2018

- Critério Todos Caminhos Linearmente Independentes ou Baseado em Complexidade: exige que todos os caminhos linearmente independentes sejam executados, determinando um limite máximo de casos de testes. Esse critério é baseado na métrica complexidade ciclomática ([MCCABE, 1976](#)).

Figura 3 – Grafo de fluxo de controle do método `cacularMDC()`

Fonte: Sérgio Luis Barbieri, 2018

- Critério Todos os Nós: exige um conjunto de casos de teste que exercitem ao menos um vez cada nó do grafo de fluxo de controle do programa;
- Critério Todos os Ramos: exige um conjunto de casos de teste que exercitem ao menos um vez cada ramo do grafo de fluxo de controle do programa.

Um possível conjunto de requisitos do critério baseado em complexidade é apresentados na Tabela 2. Isso ocorre porque pode existir mais de um conjunto de caminhos linearmente independentes. A Tabela 3 descreve os requisitos de teste para os critérios todos nós e ramos para o programa exemplo.

Assim, o conjunto de casos de teste T é adequado ao critério baseado em complexidade se os caminhos descritos na Tabela 2 forem percorridos ao menos uma vez por um caso de teste $t \in T$; é adequado ao critérios todos nós se os nós são visitados ao menos

Tabela 2 – Possível conjunto de requisitos de teste estabelecidos pelo critério baseado em complexidade para o programa exemplo.

Caminhos linearmente independentes	
1	(1,2,12)
2	(1,2,3,11,12)
3	(1,2,4,5,7,8,10,12)
4	(1,2,4,6,7,8,10,12)
5	(1,2,4,5,7,8,9,8,10,12) ou (1,2,4,6,7,8,9,8,10,12)

Fonte: Sérgio Luis Barbieri, 2018

Tabela 3 – Requisitos de teste estabelecidos pelos critérios todos os nós e todos os ramos para o programa exemplo.

Todos nós	Todos ramos
1	(2,3)
2	(3,4)
3	(4,5)
4	(4,6)
5	(8,9)
6	(8,10)
7	(2,12)
8	(3,11)
9	

Fonte: Sérgio Luis Barbieri, 2018

uma vez por pelo menos um caso de teste $t \in T$; é adequado ao critério todos ramos se os ramos são exercitados pelo menos uma vez da mesma maneira.

No apêndice A, é apresentado um exemplo de utilização da ferramenta JaCoCo para teste baseado em fluxo de controle. Esta ferramenta é utilizada para apoiar a aplicação dos critérios todos os nós e todos os ramos na indústria (MOIR, 2011).

2.5 Teste de requisitos não funcionais

Um produto de software é composto também de requisitos não funcionais, isto é, requisitos associados a restrições do software tais como requisitos de segurança, de interface e de ambiente, de desempenho entre outros. A seguir, descrevem-se alguns exemplos de testes de requisitos não funcionais.

2.5.1 Teste de desempenho

O teste de desempenho tem como objetivo submeter o sistema a determinados níveis de carga (e.g., transações, requisições, etc) com a finalidade de monitorar o consumo dos recursos computacionais (e.g., CPU, disco, tráfego de rede, etc.), permitindo desta forma validar o desempenho da aplicação (DELAMARO M. E.;CHAIM, 2010).

Teste de carga

Trata-se de uma variação do teste de desempenho, permitindo entender o comportamento da aplicação com grande volume de dados.

Teste de estresse

Esta modalidade permite entender o comportamento da aplicação em volumes acima do planejado com o objetivo de garantir que os procedimentos de recuperação de informação, recuperação de disponibilidade dos serviços e integridade de dados são tratados adequadamente.

2.5.2 Teste de usabilidade

O teste de usabilidade busca avaliar a facilidade com que os usuários interagem com o sistema. Trata-se de atividade normalmente executada com os usuários finais do sistema, a qual mede-se o tempo e a facilidade em realizar suas tarefas, comportamento de exceções etc. Os resultados deste tipo de teste podem gerar comentários para a melhoria da interação-humano computador da aplicação (DELAMARO M. E.;CHAIM, 2010).

2.5.3 Teste de segurança

Testes de segurança têm como objetivo verificar se os mecanismos de proteção introduzidos em um sistema, de fato, o protegerão de uma penetração indevida do sistema (PRESSMAN, 2006).

Este teste simula o papel do indivíduo que deseja penetrar no sistema, ou seja, quebrar as defesas que foram construídas (PRESSMAN, 2006). As atividades mais comuns são: provocar erros propositalmente esperando intrusão durante a recuperação, paralização do serviço (negação de acesso a outros) e encontrar caminhos alternativos para a entrada no sistema.

2.6 *Automatização do teste de software*

A automatização dos testes está relacionada à execução dos casos de teste (DELA-MARO M. E.;CHAIM, 2010). Os conjuntos de teste normalmente são executados todas as vezes que uma manutenção é realizada no sistema. A automatização é um procedimento que tem como objetivo facilitar a reexecução dos casos de teste com custos reduzidos, sem a necessidade de realizá-los manualmente. Em seguida serão apresentadas sucintamente duas ferramentas destinadas a este propósito.

JUnit

A ferramenta JUnit (2018) é uma biblioteca que permite escrever classes de teste. O uso consiste em codificar classes de teste nas quais os métodos são os testes que verificam as funcionalidades do sistema.

A automatização é caracterizada por possibilitar a execução encadeada de todos os testes de forma automática e ao final do processo verificar e visualizar o resultado do processo.

Selenium

O fundamento da ferramenta Selenium IDE (SELENIUM, 2018) é permitir capturar os comandos dos testadores com o objetivo de reexecutá-los de forma automática (DELA-MARO M. E.;CHAIM, 2010). Disponível para os navegadores por meio de um programa *plugin*, possibilita a automatização de teste funcional para aplicações web. Os *scripts* de testes gerados pela Selenium executam automaticamente os comandos que foram capturados durante a execução dos testes. Ao final, os resultados são comparados e um relatório é produzido.

2.7 Considerações finais

Neste capítulo, foram apresentados os diversos conceitos associados à atividade de teste como engano, defeito, erro e falha, bem como aqueles associados a estratégias e a técnicas de teste de requisitos funcionais e não funcionais.

No próximo capítulo, são discutidos os trabalhos que analisam a atividade de teste por meio de métodos qualitativos.

3 Trabalhos relacionados

Este capítulo apresenta uma revisão da literatura com o objetivo de identificar estudos realizados sobre a atividade de teste de software, utilizando pesquisas qualitativas especificamente para entender quais técnicas, métricas e ferramentas são utilizadas pela indústria de software.

3.1 Teste de software e teoria fundamentada

Esta pesquisa visa identificar um modelo de como os desenvolvedores exercem as atividades de teste no contexto de ambientes profissionais. Ela está baseada na técnica conhecida como teoria fundamentada nos dados (TF). Por isso, os trabalhos identificados nesta revisão da literatura procuram relacionar as atividades de teste com o uso de técnicas qualitativas para a sua análise.

3.2 Seleção dos trabalhos

A busca de trabalhos relacionados foi realizada nos principais repositórios de computação: *SCOPUS*¹, *IEEE XPLORE*² e *ACM DIGITAL LIBRARY*³, e também no *GOOGLE SCHOLAR*⁴. A cadeia de caracteres de busca utilizada foi “qualitative AND research AND software AND testing”. Foram encontrados 65 documentos no repositório *SCOPUS*, 159 no repositório *IEEE XPLORE* e 93 no *ACM DIGITAL LIBRARY*.

Considerando que essa pesquisa é baseada em técnica qualitativa, em especial TF, adicionalmente buscaram-se artigos relacionados a engenharia de software que fazem uso dessa técnica. A pesquisa também utilizou a cadeia de caracteres de busca “*Ground Theory*”.

Após a análise dos documentos foram identificados nove artigos relacionados à pesquisa. O Quadro 1 lista os trabalhos identificados.

---

¹ <https://www.scopus.com>

² <http://ieeexplore.ieee.org>

³ <https://dl.acm.org>

⁴ <https://scholar.google.com.br>

Quadro 1 – Trabalhos relacionados

Autor	TF	Indústria	Foco da pesquisa
Thomson, Holcombe e Simons (2009)	Não	Sim	<i>First test design</i> / TDD
Deak e Stalhane (2013)	Não	Sim	Atividades de teste
Mockus, Nagappan e Dinh-Trong (2009)	Não	Sim	Cobertura estrutural
Berlowski et al. (2016)	Não	Sim	Atividades de teste
Greiler, Deursen e Storey (2012)	Sim	Não	Atividades de teste
Coleman e O'Connor (2008)	Sim	Sim	Processo de software
Hendrick (2013)	Sim	Sim	Modelo para melhores práticas de teste
(TAIPALE; SMOLANDER, 2006)	Sim	Sim	Redução de custos atividades de teste
Cruzes et al. (2017)	Não	Sim	Teste de segurança

Fonte: Sérgio Luis Barbieri, 2018

3.3 Descrição dos trabalhos selecionados

A seguir, é apresentado a descrição dos estudos conforme o contexto de teste e prática utilizada.

3.3.1 Teste durante e ao final do desenvolvimento

Thomson, Holcombe e Simons (2009) analisaram o efeito sobre a qualidade do produto para equipes que adotam a prática TDD (*Test Driven Development*) (BECK, 2002) em relação ao teste após o desenvolvimento. Os autores utilizaram estudantes divididos em nove equipes para desenvolver três trabalhos reais da indústria, sendo que cada projeto foi desenvolvido por três equipes diferentes. As equipes selecionaram seus próprios membros, cada uma composta por três a cinco alunos de graduação de segundo ou terceiro ano.

Os dados foram coletados semanalmente, sendo que as equipes foram instruídas para enviar seus registros de teste, *script* ou documento de teste, tempo gasto em testes e códigos de programa. As equipes utilizaram as ferramentas *JUnit*⁵, *PHPUnit*⁶ e *Selenium*⁷.

O estudo concluiu que para atingir um alto nível de qualidade deve existir uma cultura de teste na equipe e foco no atendimento aos requisitos do cliente.

⁵ <https://junit.org>

⁶ <https://phpunit.de>

⁷ <http://www.seleniumhq.org>

O estudo descrito neste trabalho avalia casos reais da indústria, porém, com o foco em conhecer quais técnicas são adotadas e quais métricas são avaliadas, diferentemente deste estudo que avalia duas técnicas específicas.

3.3.2 Atividades de teste em empresas

Deak e Stalhane (2013) exploram como as atividades de teste são organizadas em empresas norueguesas e os fatores que influenciam a criação de um departamento de testes ou incentivam o investimento em pessoal dedicado a teste. A pesquisa mescla métodos qualitativos e quantitativos, utilizando como coleta de dados perguntas com múltiplas escolhas, bem como questões abertas. Em uma etapa subsequente incluiu entrevistas e uma rodada adicional de perguntas abertas. O estudo encontrou quatro categorias organizacionais nas quais as atividades de teste podem ocorrer.

Questões de pesquisa foram elaboradas para investigar a estrutura organizacional, o perfil dos testadores e o grau de automatização dos testes. Elas são descritas a seguir:

- QP1: Por que as empresas optaram por organizar testes de uma maneira particular?
  - QP1.1: Quais são as diferentes formas de organizar os profissionais de testes que podem ser encontradas em empresas de software norueguesas?
  - QP1.2: Quem é o responsável por realizar atividades de teste?
- QP2: Qual é a relação entre a organização das atividades de teste e o processo de automação de testes?
  - QP2.2: Qual é o grau de envolvimento e experiência na automatização do teste?

O estudo mostrou para as empresas que realizam atividades de teste tendem a se organizarem em quatro categorias com as seguintes distribuições:

- com departamento de teste dedicado (21,05%);
- com departamento de teste dedicado e com testadores divididos em grupos de aplicativos (10,53%);
- com testadores divididos entre os grupos de aplicativos (42,11%);
- com desenvolvedores responsáveis pelos testes, mas sem equipe de teste (26,32%).

O estudo também concluiu que em 10 das 19 empresas o responsável pelas atividades de teste é o gerente de produto ou gerente de desenvolvimento. Também a contratação de terceiros não é uma prática comum.

O resultado indica que a automatização dos testes, mesmo sendo benéfica, não é utilizada em larga escala. A prática estava presente em 14 das 19 organizações, porém, com baixa taxa de automatização:

- 25% em sete organizações;
- 50% em duas organizações;
- 75% em quatro organizações; e
- 100% em uma organização.

O estudo busca entender a estrutura organizacional das empresas em relação à equipe e ao processo de teste. O foco deste estudo é em entender como as técnicas de teste são usadas e o que determina seu uso.

[Mockus, Nagappan e Dinh-Trong \(2009\)](#) realizaram um estudo de caso múltiplo em dois diferentes projetos de software da indústria. Seus objetivos são investigar a relação entre cobertura de teste estrutural e eficácia dos testes e a relação entre esforço de teste e o nível de cobertura estrutural. As hipóteses do estudo são:

- **Hipótese 1:** O aumento do nível de cobertura diminui a taxa de defeitos.
- **Hipótese 2:** Quanto maior o nível de cobertura, cada vez é maior o esforço necessário para aumentá-la em uma dada quantidade.

Apesar das diferenças significativas entre os dois projetos do estudo, identificou-se que a cobertura de código foi associada à diminuição de problemas em campo e a uma menor probabilidade de defeitos quando esta era ajustada pelo número de mudanças antes da liberação do software. Isto sugere que a cobertura do código é uma medida sensata e prática da eficácia do teste. O estudo também descobriu que programadores mais experientes têm como tendência escrever códigos mais confiáveis e com maior qualidade, incluindo trechos para o tratamento de exceções. Essa prática diminui a cobertura, considerando que os testes normalmente não são escritos para gerar exceções.

A similaridade com este estudo está apenas no objeto da pesquisa: teste e indústria, porém, com foco exclusivo em buscar uma relação entre métrica de cobertura e falhas em campo.

3.3.3 Atividades de teste e métodos ágeis

[Berlowski et al. \(2016\)](#) realizaram uma pesquisa qualitativa exploratória com o objetivo de ampliar o conhecimento sobre teste em métodos de desenvolvimento ágeis, documentando um processo de teste de software. Para isso, utilizou um projeto real na indústria desenvolvido para uma empresa do segmento de telecomunicação. O artigo descreve como os desafios são superados, de que maneira eles afetam o processo de teste e quais os resultados obtidos.

O estudo é focado no processo de teste com requisitos para versões não programadas e infraestrutura complexa, avaliando o nível de adoção de práticas de testes, tais como desenvolvimento orientado por testes (TDD), automação de testes, programação em pares, entre outras. As questões de pesquisa foram:

- QP1: Até que ponto o requisito para “versões” (*releases*) não programadas é atendido?
- QP2: Como é realizada a automação de teste?
- QP3: Quanto esforço de automatização de teste é necessário (em relação a diferentes tipos de atividades e ferramentas)?

Como contribuição, os autores fornecem um estudo de caso detalhado para os profissionais interessados em implementar um processo de teste ágil em um ambiente similar. Também avalia o uso de ferramentas de automatização de teste e as dificuldades encontradas na implementação. O estudo fornece uma breve descrição de todas as ferramentas que foram usadas para implementar testes automatizados, prontas para uso, possibilitando auxiliar outros profissionais que enfrentam desafios semelhantes.

O estudo tem como alvo a indústria, porém, em uma empresa específica e método de desenvolvimento ágil com o objetivo de avaliar o atendimento de versões não programadas e automatização de teste.

3.3.4 Teste de aplicativos plugin

[Greiler, Deursen e Storey \(2012\)](#) conduziram um estudo qualitativo utilizando a técnica teoria fundamentada, entrevistando 25 profissionais seniores sobre como eles testam

aplicativos *plug-in* com base na arquitetura de *plug-in* do ambiente de desenvolvimento Eclipse⁸.

Uma das questões de pesquisa (QP4) foi descobrir se existiam estratégias especiais para o teste de *plug-ins*. Por meio da TF, foram identificadas três importantes estratégias:

- Auto-hospedagem de projetos. Os desenvolvedores são usuários do próprio projeto, utilizando-o internamente como ferramenta, facilitando, por isso, o relato de problemas.
- Envolvimento do usuário. Os usuários também são desenvolvedores e quanto maior o número de instalações do projeto, maior será a quantidade de formas diferentes que o produto é utilizado (e.g., versões, sistema operacional). Dessa forma a quantidade de testes realizadas pelos responsáveis em relação a como o produto é utilizado na prática é maior.
- Envolvimento do desenvolvedor. A arquitetura de *plug-in* do Eclipse permite aos desenvolvedores criar *plug-ins* a partir de outros *plug-ins*. Por isso, os usuários do software são muitas vezes desenvolvedores qualificados cujos projetos também dependem da qualidade dos outros projetos. Como consequência, em projetos próprios dedicam parte do seu tempo para melhorar projetos dependentes.

A similaridade deste estudo com esta pesquisa está no uso da técnica TF como ferramenta de coleta. Apesar de o estudo estar relacionado com teste, ele é específico para um artefato do ambiente de desenvolvimento Eclipse: *plug-ins*.

3.3.5 Melhoria da qualidade de software

Coleman e O'Connor (2008) realizaram estudo qualitativo para entender o processo de melhoria da qualidade de software na indústria. A metodologia escolhida para o estudo foi a TF. Como ferramenta de coleta de dados, o estudo elaborou entrevistas semi-estruturadas com as equipes de projetos envolvidas no estudo. Possui similaridade com o estudo realizado na coleta de dados por meio de entrevistas, transcrição e análise com a técnica de TF.

Segue abaixo o resumo das questões de pesquisa com a respectiva conclusão:

---

⁸ <https://www.eclipse.org/>

- Quais processos de software estão sendo utilizados por empresas desenvolvedoras de software?

A pesquisa revelou que nenhuma empresa está utilizando um modelo de processo inovador, porém, todas estão utilizando algum tipo de processo proprietário baseado em maior ou menor grau em modelos padrão.

- Como o processo é formado ou criado?

Os resultados mostram que isso depende de vários fatores. O principal fator refere-se ao conhecimento do gestor de desenvolvimento. A experiência acumulada pelo profissional encarregado de gerenciar determina como será o início do processo de desenvolvimento.

- Como e por que os processos de desenvolvimento mudam?

Segundo os dados analisados, não há indícios de iniciativa dos profissionais no que diz respeito a mudanças em seus processos de desenvolvimento. A maioria dos entrevistados apresentaram-se satisfeitos com os seus processos atuais e, enquanto funcionam, não fariam ajustes por medo de “quebrar algo”. Isto significa que a melhora do processo é, em essência, reativa.

- Por que as empresas de software não estão utilizando as “melhores práticas” dos *frameworks* de desenvolvimento?

O estudo concluiu que a implementação e a manutenção de qualquer iniciativa de melhoria de processo incorrem em custos significativos. Os recursos necessários para executar aperfeiçoamento de fluxo são proporcionalmente muito maiores em empresas de menor porte. Essas empresas têm como prioridade a própria sobrevivência no mercado e, em seguida, estabilidade em relação a melhoria de processo.

3.3.6 Uso de teoria fundamentada em teste de software

[Hendrick \(2013\)](#) realizou estudos qualitativos baseados em TF para avaliar as atividades de teste no ciclo de desenvolvimento. O problema de pesquisa investigado foi explorar se a TF pode ser usada para desenvolver uma estrutura para melhores práticas em teste de software. A pesquisa foi aplicada em uma empresa específica do segmento de telecomunicações localizada em Dublin, Irlanda. Como método de coleta, etapa da TF, o estudo elaborou sete questões para ser utilizada em entrevistas estruturadas aplicadas

em cinco profissionais de teste com experiência média de sete anos. Após a transcrições, as mesmas foram utilizadas como entrada para o programa *Atlas.TI*. Como resultado da análise, o projeto elaborou um “framework” de melhores práticas. Esta pesquisa é semelhante na metodologia, porém, difere nos objetivos e resultados esperados. A busca das melhores práticas de teste está baseada em cenário muito particular: empresa de telecomunicação, estrutura organizacional e produtos, o que difere dos objetivos desse estudo.

3.3.7 Complexidade do teste

[Taipale e Smolander \(2006\)](#) elaboraram pesquisa qualitativa para compreender a complexidade prática de teste, com o objetivo de gerar propostas para reduzir custos de desenvolvimento e teste, bem como melhorar a qualidade de software. Eles utilizaram a TF como método de pesquisa. O estudo foi realizado em 26 unidades organizacionais do segmento de telecomunicações localizadas na Finlândia. Os dados foram coletados por meio de entrevistas estruturadas e semiestruturadas. O objetivo das entrevistas, em uma primeira rodada, foi entender as melhores práticas de teste, identificar os problemas e as propostas de melhoria. Outras entrevistas foram realizadas para um maior aprofundamento das práticas de teste da complexidade em testar software e componentes e da automatização de teste.

Como resultado final, o estudo apresentou propostas de melhoria de processo. Em comparação com este estudo, o foco principal da pesquisa de [Taipale e Smolander \(2006\)](#) é o processo de desenvolvimento e atividades correlatas, por exemplo, a comunicação de equipes e os papéis e responsabilidades dos profissionais do projeto.

3.3.8 Testes de segurança

[Cruzes et al. \(2017\)](#) conduziram uma pesquisa qualitativa sobre como testes de segurança são realizados em projetos de software utilizando metodologia ágeis. O trabalho definiu três questões de pesquisas e utilizou a técnica de coleta de dados por meio de entrevistas semiestruturada. O escopo de coleta foram quatro equipes de desenvolvimento. O objetivo foi gerar diretrizes de melhores práticas. As questões de pesquisas foram:

- QP1: Como o processo de engenharia de segurança tradicional é gerenciado e organizado nas equipes ágeis?
- QP2: Como as equipes ágeis realizam testes de segurança em cada fase de teste?
- QP3: Como as técnicas de teste de segurança tradicionais geralmente são usadas no ciclo de vida de desenvolvimento de software ágil?

Os principais resultados obtidos foram os seguintes. Com relação à questão QP1, observou-se que as grandes empresas possuem *chief security officer* (CSO) próprio, mas ele não faz parte da equipe de desenvolvimento para não interferir nas atividades diárias. Notou-se também que para empresas pequenas não existe a hierarquia ou a definição de um CSO. Observou-se que testes de penetração normalmente são realizados por terceiros.

Além disso, o resultado de QP2 mostrou como resultado que é comum o uso de teste de unidade, porém, não sendo a segurança o foco dos testes ou a consideração de aspectos específicos de segurança, os quais são tratados no teste de sistema.

A partir de modelos e abordagens clássicas de teste de segurança, os entrevistados foram questionados para comentar como executam as atividades na metodologia ágil com o objetivo de responder a QP3. A conclusão foi que não há um modelo específico para segurança, apenas abstrações para as questões de segurança. Outro ponto citado foi a utilização de terceiros para teste de penetração.

3.4 Considerações finais

Os trabalhos apresentados mostram que técnicas qualitativas de pesquisa têm sido utilizadas para investigar a atividade de teste em diferentes contextos. Elas foram usadas, por exemplo, para caracterizar o teste nos métodos ágeis, o teste de segurança e o papel da cobertura de código na detecção de defeitos, entre outras questões.

Este trabalho utiliza métodos semelhantes, em particular teoria fundamentada, para identificar um modelo de execução da atividade de teste em organizações brasileiras de desenvolvimento de software. Diferentemente dos trabalhos apresentados, a pesquisa realizada tem como objetivo determinar os fatores que determinam a adoção de técnicas e estratégias de teste, isto é, verificar como a atividade de teste dá-se na prática.

No capítulo a seguir, são apresentados os procedimentos metodológicos para condução da pesquisa qualitativa.

4 Procedimentos metodológicos

O objetivo da pesquisa é entender como os profissionais da indústria realizam as atividades de teste. A pesquisa foi realizada com uma abordagem qualitativa, na qual os resultados não podem ser obtidos por meio de procedimentos estatísticos. Entre os diversos métodos existentes, o estudo é baseado na metodologia *teoria fundamentada* (STRAUSS A.;CORBIN, 2008). A teoria fundamentada significa que a teoria foi construída a partir de dados, sistematicamente reunidos e analisados por meio do processo de pesquisa. Essa metodologia foi desenvolvida por dois sociólogos, Barney Glasser e Anselm Strauss, entre os anos de 1967 e 1992.

Essa abordagem qualitativa é definida por três componentes fundamentais: dados, procedimentos e codificação. Os *dados* podem ser obtidos por meio de questionários, entrevistas ou observações. Os *procedimentos* são técnicas utilizadas pelos pesquisadores para interpretar e organizar os dados. A *codificação* consiste em conceitualizar, reduzir, elaborar e relacionar os dados, sendo o passo inicial para a construção da teoria por meio de conceitos identificados. Em síntese, por meio de métodos para a coleta de dados, consegue-se identificar conceitos e explicações sobre fenômenos.

A execução da pesquisa baseada na teoria fundamentada pode ser separada em três etapas: *coleta de dados*, *codificação* (ou categorização) e *redação da teoria*. A coleta de dados é feita por meio de observação dos participantes, entrevistas e questionários. Nessa metodologia, as atividades de análise são executadas simultaneamente até ocorrer a saturação teórica, indicando que novos dados não serão descobertos. O objetivo dessa etapa é coletar o maior volume de informações. O procedimento de rotular e analisar os dados coletados é denominado de codificação. A codificação consiste na aplicação de técnicas de análise minuciosa dos dados para o desenvolvimento e formação de uma teoria. A teoria consiste em um conjunto de conceitos bem desenvolvidos relacionados por meio de declarações de relações que, juntas, constituem uma estrutura integrada que pode ser usada para explicar ou prever fenômenos (STRAUSS A.;CORBIN, 2008).

A seguir, serão detalhados os conceitos e os processos associados à metodologia teoria fundamentada utilizada do desenvolvimento desta pesquisa.

4.1 Coleta de dados

Conforme Robson (2011), há vários métodos disponíveis para coleta de dados, sendo que os mais usuais são entrevistas, questionários e observação direta. A escolha do método também pode depender de diversos fatores e condições específicas de cada projeto, por exemplo, custo e tempo.

Robson (2011) sugere uma regra simples para a seleção dos métodos:

- para descobrir o que as pessoas fazem em público, utilize observação direta;
- para descobrir o que eles fazem em particular, utilize entrevistas ou questionários;
- para descobrir o que eles pensam, sentem ou acreditam, utilize entrevistas, questionários ou escala de atitude; e
- para determinar suas habilidades ou mensurar suas inteligências ou personalidade, utilize testes padronizados.

Para esta pesquisa, foram utilizados entrevistas. Questionários e entrevistas são popularmente utilizados para pesquisas (*surveys*), mas também como método primário e secundário de coleta de dados. Existem diversas abordagens para a aplicação de um questionário: autopreenchimento, via entrevista, por telefone e via Internet. Cada uma delas exige atenção especial na sua elaboração em relação à complexidade, distribuição (público-alvo), ao uso de questões abertas e ao tamanho.

Ainda sobre a aplicação, a qualidade dos resultados obtidos pode variar sensivelmente, dependendo da ordem em que as questões serão respondidas, do uso de recursos visuais, etc. As perguntas devem ser projetadas para ajudar a alcançar os objetivos da pesquisa. Os entrevistados devem entender as questões da maneira que o pesquisador planejou, devem ter acesso à informação necessária para respondê-las e devem estar dispostos a respondê-las na forma solicitada.

Abaixo segue uma lista de verificação para ajudar a evitar problemas na redação de perguntas extraída e adaptada de Robson (2011):

- manter a linguagem simples;
- manter questões curtas; questões longas ou complexas são difíceis de entender;
- evitar duas questões em uma;
- evitar questão principal;

- evitar questão em sentido negativo; e
- perguntar apenas o que os entrevistados tenham conhecimento necessário para responder.

A entrevista, quando usada como método de pesquisa, conta com o pesquisador realizando perguntas e recebendo as respostas da pessoa que está sendo entrevistada. Segundo [Robson \(2011\)](#), este é um método largamente utilizados em pesquisas sociais.

Há diferentes tipos de entrevistas, variando na sua forma de conduzir a conversa e de obter informações, classificadas em três modelos, sendo eles: pesquisa estruturada, semi-estruturada e não-estruturada. A pesquisa estruturada parte de questões pré-determinadas, sendo um tipo de entrevista mais rígido. Apesar de as perguntas serem já estipuladas e definidas, as respostas são abertas e, assim, não limitam a fala do entrevistado.

Já o tipo semiestruturado possui um guia de entrevista, com uma lista de tópicos para cobrir por completo todos os assuntos pertinentes. Mesmo com a pauta estipulada, a ordem pode ser modificada de acordo com o ritmo da entrevista e com as respostas que aparecem ao decorrer. Pode-se também adicionar tópicos extras quando surgir a necessidade. Por fim, a pesquisa não-estruturada é completamente informal, podendo ser considerada uma conversa entre entrevistador e entrevistado. Aquele que conduz a entrevista tem uma ideia ampla do que vai ser falado, mas não segue uma ordem ou guia de perguntas.

Independente de qual tipo de entrevista é feita, [Robson \(2011\)](#) lista algumas práticas para serem evitadas na formulação das perguntas. Não fazer questões longas é uma delas, uma vez que o entrevistado não costuma lembrar da pergunta por completo e assim não responde integralmente. Outra recomendação é adequar a linguagem ao público alvo, evitando jargões e termos técnicos que dificultem o entendimento da pergunta. Não induzir o seu entrevistado é também um ponto essencial para obter um resultado imparcial, assim como evitar perguntas direcionadas para um único viés, que possam deixar a sua pesquisa unilateral.

Partindo do princípio de que as ações e comportamento das pessoas é um aspecto central que guia a maioria das pesquisas no mundo real, podemos utilizar a observação como técnica simples e natural para registrar o que acontece e, então, descrever, analisar e interpretar tudo que foi percebido ([ROBSON, 2011](#)).

A observação direta é um método para obter dados e que, assim, completa as demais técnicas já apresentadas, tais como questionário e entrevista. Dessa forma consegue-se assegurar um resultado mais preciso e eliminar as discrepâncias deixadas pelos outros métodos.

[Robson \(2011\)](#) categoriza quatro formas de participação, sendo elas: participante completo, participante como observador, participante marginal e observador como participante. O participante completo trabalha agindo naturalmente, fazendo parte daquele grupo. É como um espião infiltrado que toma nota do que acontece ao seu redor. Já o participante como observador deixa claro qual o seu papel no grupo, podendo perguntar e questionar determinados acontecimentos que se passam durante a análise e, neste caso, é importante ter a confiança dos membros do grupo.

Já se tratando de participação marginal, acontece um envolvimento menor do participante durante a sua análise. Ele está inserido no contexto, porém, se comporta de forma passiva. Neste tipo de observação é necessária muita atenção e, se possível, tomar nota ao longo da experiência. Também é interessante atentar-se às roupas utilizadas e comportamento, para que não se destoe do grupo. Alguns participantes marginais são muito próximos dos observadores completos, aqueles que não participam da atividade e não assumem o seu papel como pesquisador.

E, por fim, o observador como participante não interfere nas atividades que estão sendo executadas. Apesar de ter essa atribuição discreta, todos aqueles que estão envolvidos na atividade sabem do seu cargo como pesquisador.

4.2 Codificação

A coleta de dados é realizada utilizando técnicas específicas tais como questionários, entrevistas e observações. A etapa subsequente é a análise dos dados, processo denominado de codificação.

A codificação consiste em aplicar técnicas e procedimentos para identificação de categorias e propriedades. Ferramentas analíticas são mecanismos e técnicas utilizadas por analistas para auxiliar o processo de codificação ([STRAUSS A.;CORBIN, 2008](#)). O objetivo é evitar pré-concepções, estimular o pensamento indutivo, ouvir as pessoas, focar nos dados e descobrir as propriedades e dimensões das categorias.

Como exemplo de ferramentas analíticas, pode-se citar o questionamento, a análise de palavra, frase ou parágrafo e a análise por meio de comparações. O uso do questionamento consiste em elaborar boas perguntas para a melhoria do desenvolvimento da teoria, ajudando o analista a desbloquear as análises. A análise de uma palavra, uma frase ou um parágrafo busca estabelecer possíveis significados, melhorando a interpretação da análise, devido à interpretação derivada do uso cultural comum ou da experiência.

A análise por meio de comparações, segundo [STRAUSS A.;CORBIN \(2008\)](#), é essencial para identificar categorias e para desenvolvê-las. As comparações podem ser classificadas em dois grupos:

Comparação de incidente por incidente: busca similaridade e diferenças entre suas propriedades para classificação;

Comparações teóricas: compara categorias em busca de conceitos similares ou diferentes para revelar possíveis propriedades e dimensões quando elas não são evidentes para o analista.

Os processos de codificação são classificados em codificação aberta, codificação axial e codificação seletiva. Não há uma ordem previamente definida em termos de início e fim, ou seja, os processos podem ser executados em paralelo à medida que novos dados são coletados. Nas subseções abaixo são descritos cada um dos processos.

4.2.1 Codificação aberta

Para entender de forma aprofundada o que é de fato a codificação aberta é necessário compreender os termos que a cercam.

Os *fenômenos* são as ideias centrais nos dados e são representadas como conceitos. Já os *conceitos* são os pilares que formam a teoria. As *categorias* são os próprios conceitos que representam os fenômenos e que são caracterizadas pelas *propriedades*. Dentro das propriedades, tem-se as *dimensões*, que especificam as categorias e as subcategorias, que ainda são conceitos que pertencem à categoria e acrescentam especificações.

[STRAUSS A.;CORBIN \(2008\)](#) apresentam um exemplo simples desse processo: se uma pessoa observa 10 objetos no céu e os rotula como “pássaros”, depois observa cinco objetos diferentes e os rotula como “aviões”, e depois observa mais sete objetos e os chama

de “pipas”, mais cedo ou mais tarde ela pode se perguntar o que esses objetos têm em comum e chegar ao conceito de “vôo”. Esse termo não apenas permite aos objetos serem classificados, mas também explica o que eles estão fazendo (em termos de ação).

Tendo o conhecimento necessário sobre os termos que formam a codificação aberta, pode-se seguir para a criação da teoria propriamente dita. São estabelecidos os conceitos, definidas as categorias e desenvolvidas as propriedades e suas dimensões e, se necessário, criadas também subcategorias.

A conceituação é um processo claro e linear que permite distinguir itens diferentes e agrupar aqueles que são semelhantes entre si, seguindo suas propriedades e dimensões. Também faz parte o ato de nomear essas categorias de maneira que represente a semelhança que as unem. E, quando se tem já algumas categorias, especificam-se as suas propriedades. Os conceitos sofrem alterações de acordo com as propriedades. Observam-se também padrões de comportamento, que são analisados a partir de especificações como as dimensões. Com dados, é criada uma base forte para a construção da teoria.

4.2.2 Codificação axial

A codificação axial é a etapa seguinte no fluxo de codificação. Ela se baseia no processo de conectar categorias às suas devidas subcategorias. O seu nome é justificado pelo elo entre uma categoria e os níveis de propriedades e dimensões. Ou seja, tudo gira em torno de uma categoria. Os demais elementos que se deve tomar nota do seu significado quando colocados neste contexto são: paradigma, estrutura e processo. O paradigma é uma ferramenta de análise. Para que se possa agregar estrutura com processo, o paradigma se faz absolutamente necessário. A estrutura situa a categoria (fenômeno), isto é, fica responsável por fornecer um contexto que é capaz de especificar ainda mais a categoria em que é referida. O processo nada mais é que uma série de ações relacionadas ao fenômeno e à passagem de tempo. O ato de codificar em torno de uma categoria, partindo de um único eixo, fornece maior profundidade à análise. Desenvolve-se as categorias e então procura-se um meio de conectá-las, a partir daí parte-se para a construção da teoria. Deve-se levar em consideração que o paradigma nada mais é do que uma ferramenta de análise, é um instrumento para alcançar novos resultados e dados e não uma verdade absoluta que fornece uma única resposta sobre o que é observado. Para os analistas não há um único

momento em que os conceitos se relacionam. As pequenas conexões vão fazendo sentido em horas inesperadas, até que o quebra-cabeça encaixa-se por inteiro. Não há um tempo determinado para que se chegue a todos os conceitos relacionados, mas sim períodos de tempo em que se percebe o elo entre os conceitos e as conclusões acontecem de pouco em pouco.

4.2.3 Codificação seletiva

Por fim, chegamos à codificação seletiva. Depois das fases de codificação aberta e axial, inicia-se o processo de integração e refinamento da teoria. Há duas expressões que serão de extrema importância para evoluir esta codificação. A primeira delas é a saturação teórica. A fase da saturação teórica faz parte do processo e cabe aos analistas identificarem o momento em que não aparecem mais propriedades, dimensões ou relações na análise. Já a segunda expressão, é a de limite de variabilidade. Diz respeito ao nível de variação das dimensões em relação às suas propriedades, estabelece-se a variação por meio de amostragem. A codificação seletiva configura-se como etapa final devido aos seus dois elementos chaves apresentados acima: saturação teórica e limite de variabilidade. Neste ponto, os dados necessários para o refinamento da teoria estão disponíveis; ocorre então a integração dos dados iniciais da pesquisa até as conclusões finais. Com a teoria já bem integrada, é hora de refiná-la. Para que possa ser validada, é preciso ser associada aos dados brutos e reconhecida pelos participantes. Os conceitos devem ser reconhecidos por eles, pelo menos de forma geral, para que possam ser confirmados.

4.3 Projeto

O projeto consiste em pesquisa qualitativa baseada em teoria fundamentada realizada em empresas do segmento de desenvolvimento de sistemas. A execução do projeto foi realizada em entrevistas semiestruturadas baseado no processo da teoria fundamentada.

4.3.1 Projeto para coleta de dados

A Figura 4 ilustra os procedimentos para a execução da pesquisa. O trabalho será realizado em três grandes etapas, sendo que a primeira e a segunda etapa destinam-se à coleta e à análise sistemática de dados, aplicando-se as técnicas da teoria fundamentada para determinar as categorias e a saturação teórica.

Figura 4 – Processo de pesquisa teoria fundamentada

Fonte: Sérgio Luis Barbieri, 2018

Para atender a etapa de coleta de dados, foram desenvolvidos questionário e guia de entrevista. Foram aplicados duas formas de coleta de dados para as equipes participantes do projeto. O Quadro 2 apresenta o guia de entrevista, que busca coletar o máximo de informações de forma livre sobre as práticas e técnicas utilizadas pela empresa. Ele foi utilizado para os profissionais responsáveis pela coordenação e liderança de equipes de desenvolvimento.

Quadro 2 – Guia Entrevista

Item	Questão
1	Qual é a melhor definição para a sua empresa? (<i>empresa com equipe própria para desenvolvimento de sistema; empresa com equipe para desenvolvimento de sistema terceirizada; empresa prestadora de serviço em desenvolvimento de sistema; empresa prestadora de serviço em desenvolvimento de sistema especializada em teste de software; outra</i>)
2	Quais técnicas de teste são utilizadas pela empresa? (<i>funcional; estrutural; baseada em defeitos; desempenho; sistema; outras</i>)
3	Como é definido o escopo de teste?
4	Qual é o critério de parada dos testes?
5	Quais ferramentas de teste são utilizadas pela equipe?
6	Ferramentas de cobertura de código são utilizadas pela empresa?
7	Os testes são automatizados?
8	Como os resultados dos testes são validados? Há critérios de aceitação?
9	Quem é o responsável pela validação ou aceitação?

Fonte: Sérgio Luis Barbieri, 2018

4.3.2 Análise de dados

Para a análise dos dados foi utilizada a ferramenta [ATLAS.ti Scientific Software Development GmbH \(2017\)](#), versão de avaliação. A ferramenta é específica para análise qualitativa; ela permite a inserção das transcrições, codificação, geração de categorias, agrupamento das categorias, inserção de lembretes, relacionamentos, contagens de palavras e ocorrências etc. Possui uma diversidade de relatórios e exportação de dados, auxiliando no processo de execução da metodologia.

4.4 Considerações finais

Este capítulo apresentou sucintamente a técnica de pesquisa qualitativa teoria fundamentada, descrevendo como os dados são coletados, os protocolos que foram utilizados

nas entrevistas e como os dados foram analisados. No próximo capítulo será apresentado os resultados da pesquisa.

5 Resultados do estudo qualitativo

Neste capítulo, será apresentado o resultado da pesquisa qualitativa com a técnica TF aplicada em cinco empresas. A técnica foi conduzida por meio de entrevistas com cinco profissionais.

A seguir, será apresentada a descrição das empresas e os resultados obtidos.

5.1 Descrição do processo de seleção das empresas e dos indivíduos entrevistados

Foram selecionadas empresas de diversos segmentos e estrutura organizacional. Todas as empresas, no momento das entrevistas, possuíam projetos em andamento. O Quadro 3 ilustra as empresas participantes. A *Empresa A* desenvolvia um software para propostas de seguro com múltiplas funcionalidades (e.g., orçamento, endosso, renovação etc.). A *Empresa B* possui software destinado à integração bancária, chamado de frente de caixa. O sistema combina tecnologias de interface gráficas web e componentes de integração com o sistema de retaguarda (*backoffice*) de terceiros. A solução é customizada para cada cliente em razão das particularidades de cada instituição bancária. A *Empresa C*, por ser especializada na prestação de serviços de teste, possui um amplo histórico de projetos, destacando-se em sistemas especializados em transações bancárias e cartão de crédito.

Já a empresa *Empresa D* desenvolve sistema destinado à avaliação e concessão de crédito, sendo seus principais clientes instituições bancárias e financeiras. O sistema passa por constantes customizações para adequação aos requisitos de cada cliente. Finalmente, a empresa *Empresa E* desenvolve e mantém um produto destinado ao ramo imobiliário comercial (*real estate*). O sistema possui funcionalidades de contratos, controle de obras, administração de condomínios, entre outras. As entrevistas foram aplicadas aos profissionais responsáveis pela atividade de teste, sendo dois gerentes de projeto, dois coordenadores e um analista de teste. As entrevistas foram realizadas seguindo o protocolo descrito no capítulo anterior. Os entrevistados estão demonstrados no Quadro 4. Todos os profissionais têm expressiva vivência na área de teste.

Quadro 3 – Empresas participantes x segmento

Identificador	Segmento	Descritivo
Empresa A	Seguro	Posicionada entre a maiores empresas do Brasil na categoria seguros gerais.
Empresa B	Automação Bancária	Soluções bancárias de automação e integração
Empresa C	Prestador de serviço	Empresa especializada em testes; atende grandes clientes em diversos segmentos com destaque para a área bancária, cartão de crédito e telefonia
Empresa D	Financeiro	Comercializa sistemas com produtos exclusivos para avaliação e concessão de crédito. Grandes instituições financeiras são clientes.
Empresa E	Setor imobiliário	Empresa de médio porte; presta serviço de desenvolvimento para a área imobiliária comercial (<i>real estate</i>), provedor de comunicação e serviços de <i>datacenter</i> .

Fonte: Sérgio Luis Barbieri, 2018

Quadro 4 – Profissional x Experiência

Empresa	Profissional	Experiência em teste
C	Gerente de projeto	10 anos
E	Gerente de projeto	15 anos
B	Coordenador	5 anos
D	Coordenador	3 anos
A	Analista de teste	4 anos

Fonte: Sérgio Luis Barbieri, 2018

5.2 Descrição do processo de coleta de dados

As entrevistas semiestruturadas foram utilizadas como instrumento de coleta de dados. Elas foram gravadas e posteriormente transcritas em documento eletrônico. Como etapa final deste processo, os documentos foram inseridos na ferramenta [ATLAS.ti Scientific Software Development GmbH \(2017\)](#). Ela permite unificar todos os documentos em um único ambiente e por meio de suas funcionalidades registrar a análise dos dados. As próximas seções descrevem os passos da análise.

5.3 Descrição do processo de análise de conteúdo

As transcrições foram analisados com o apoio da ferramenta Atlas.ti. A partir das transcrições inseridas na ferramenta, inicia-se a atividade de codificação aberta, cujo objetivo é identificar as categorias. A identificação das categorias é a primeira etapa da TF. Neste processo de codificação a ferramenta permite a inclusão de “comentários” e “lembretes”. À medida em que novos documentos são codificados, inicia-se também a

codificação axial, processo que consiste no refinamento das categorias: junção de categorias, reclassificação, entre outros.

Na etapa seguinte, dá-se a codificação seletiva. Nesta etapa, as categorias são agrupadas e relacionadas em um nível mais abstrato, surgindo os conceitos e a categoria principal, cujo objetivo é formar a teoria. O resultado final do processo é apresentado no Quadro 5.

Quadro 5 – Conceitos e categorias

Conceito	Categoria
Modelo organizacional	Organização do trabalho Desenvolvimento de sistemas com equipe própria Porte da organização Prestador de serviço em teste de sistemas
Modelo e estrutura do projeto	Orçamento Imagem da empresa Prazo Processo de desenvolvimento de Software Requisitos funcionais Tipo de sistema
Categoria principal	Categoria
Abordagem de teste	Condução do teste Método de desenvolvimento Teste funcional Teste não funcional Volume de teste

Fonte: Sérgio Luis Barbieri, 2018

5.4 Resultados da análise de conteúdo

Conforme conceituado no Capítulo 4, a teoria é construída a partir de dados, sistematicamente reunidos e analisados, construindo um quadro teórico a partir dos relacionamentos. Os conceitos, categorias e relacionamentos identificados pela pesquisa são apresentados na Figura 5.

Observe-se que na Figura 5 há elementos de diferentes cores. Os retângulos brancos representam o agrupamento de categorias que definem uma determinada teoria. As cores para os demais retângulos foram utilizadas para destacar os grupos de categorias e subcategorias associadas à teoria.

As linhas tracejadas na cor vermelha significam a *conexão* entre categoria e teoria, já as linhas sólidas na cor preta significam a *associação* da subcategoria com a categoria. As associações podem ser das seguintes formas: “*está associada a*”, “*é parte de*” ou “*é causa de*”.

Nas próximas seções, serão apresentados em detalhe os resultados obtidos.

5.4.1 Modelo organizacional

O conceito *Modelo organizacional* destacado na Figura 6 das empresas em síntese define como será a organização do trabalho: prestadores de serviços, papéis e responsabilidades. O porte da organização normalmente direciona a formação da equipe de desenvolvimento: própria, mista ou terceiros. Em equipes próprias, a possibilidade de integração das etapas de desenvolvimento e principalmente das atividades de teste é um facilitador, apesar de não haver evidências consistentes sobre esse prática nos dados coletados.

A integração observada foram apenas no compartilhamento dos requisitos funcionais, base para elaboração dos casos de teste. Em empresas com o processo verticalizado (equipe própria em todas as etapas do desenvolvimento), há uma definição prévia do plano de teste que guia a elaboração dos casos de teste e as técnicas a serem aplicadas. O entrevistado da **Empresa D** afirmou: “*Hoje a gente recebe o software e juntamente com ele uma documentação com um plano de teste que tem passos a seguir...*”.

Independente do modelo organizacional, constatou-se que os testes são reexecutados pelo contratante, porém, com um escopo menor. O entrevistado da **Empresa B** comentou: “*O cliente também testa...passa pelo teste deles que é a homologação...*”.

5.4.2 Modelo e estrutura do projeto

O conceito *modelo e estrutura do projeto* está fundamentado em três importantes categorias: orçamento, prazo e tipo de sistema. O “orçamento” é decisivo para a escolha das técnicas e ferramentas de teste. O tempo e o grau de cobertura das atividades de teste são planejados conforme o orçamento disponível. O que o estudo observou foi a priorização em testar funcionalidades “chaves” do sistema, seja por atendimento ao negócio, seja

Figura 6 – Modelo organizacional

Fonte: Sérgio Luis Barbieri, 2018

para manter a imagem da empresa. A imagem da empresa está relacionada à repercussão negativa na ocorrência de falhas em campo, devido a defeitos que permaneceram no software, problemas de desempenho ou problemas de segurança. O conceito está destacado na Figura 7,

O “tipo de sistema” é outro fator que determina uma técnica ou tipo de teste. Por exemplo, sistemas concebidos no modelo *web* tendem a utilizar testes funcionais; já sistemas baseados em *Application Programming Interface* (API), o estudo observou que a preocupação maior é o desempenho, sendo assim, uma boa parte do orçamento e tempo dedicado a este requisito não funcional.

A categoria “prazo” tem relação direta à automatização de teste. A preocupação em automatização não está associada à facilitação do teste em futuras manutenções ou implementações de novas funcionalidades, mas ao atendimento a prazos estabelecidos. Se

Figura 7 – Modelo e estrutura do projeto

Fonte: Sérgio Luis Barbieri, 2018

uma determinada funcionalidade possui uma grande combinação de entrada e a mesma for consumir muito recursos para a execução de forma manual, ela é uma candidata à automatização.

5.4.3 Abordagem de teste

O conceito “abordagem de teste” é bem amplo e diversificado por conter várias categorias e subcategorias relacionadas a esse conceito conforme destacado na Figura 7. Uma das categorias de maior relevância é a “condução do teste”, associada à gestão do teste. Ela é a atividade de maior relevância para o gestor. Há uma grande preocupação em documentar a condução das atividades de teste, principalmente em criar evidências dos

Teste funcional x teste estrutural

Conforme descrito na seção anterior, a escolha de técnicas de teste não está diretamente relacionada em utilizar a melhor técnica ou ferramenta, mas no cumprimento da etapa de teste com os respectivos registros: relatórios e evidências. Sendo assim, a definição do plano de teste não leva em consideração uma discussão mais profunda sobre o tema.

Também foi identificado que a tarefa de teste é executada por terceiros que fazem parte da equipe de projeto, porém, seguindo os padrões do contratante e muitas vezes utilizando as ferramentas e controles definidos pelo contratante.

Automatização

Conforme citado acima, a automatização está relacionada apenas à redução do esforço manual em etapas específicas do teste, a qual é claramente percebida uma repetição e impossibilidade de execução por pessoas. O estudo também evidenciou outras situações relacionadas a esse tema:

- falta de conhecimento das equipes em ferramentas de automatização;
- falta de prática no uso das ferramentas, gerando aumento da carga de trabalho na concepção do software;
- preocupação em cumprir o escopo e prazo apenas do projeto atual, sem a visão em utilizar os casos de teste automatizados em ciclos futuros (e.g., manutenção).

Teste não funcional x ferramentas

No estudo, as ferramentas de teste citadas (e.g., JMeter¹) estão relacionadas a testes de desempenho e carga cujo objetivo é atender requisitos não funcionais. Requisitos não funcionais normalmente são executados com a apoio de ferramentas que já possuem funcionalidades intrínsecas de automatização. O entrevistado da **Empresa C** citou exemplos de segmentos e funcionalidades nas quais o tempo de resposta é a maior preocupação (e.g., transações bancárias, cartão de crédito, *e-commerce*) para a homologação do software.

---

¹ <http://jmeter.apache.org/>

As ferramentas de teste de requisitos não funcionais são úteis para testes de sistemas que utilizam computação em nuvem e fazem uso de APIs. Uma relação mais profunda entre arquitetura de sistemas e técnicas de teste não foi abordada por essa pesquisa.

5.5 Teoria resultado da aplicação da análise fundamentada

O ponto principal da teoria fundamentada é a geração da teoria a partir do desenvolvimento e relacionamentos das categorias com o objetivo de formar um quadro teórico. Baseado na análise dos resultados discutidos nas seções anteriores, o estudo chegou a seguinte teoria em relação a como as empresas conduzem as atividades de teste e quais os fatores que influenciam ou determinam o teste nas empresas participantes.

A abordagem de teste é dependente do modelo organizacional que é responsável por definir a estrutura do projeto na formação da equipe. A formação da equipe é determinante na organização e condução dos trabalhos. A organização dos trabalhos, normalmente constituída por prestadores de serviços, direciona a adoção de modelos e técnicas básicas de teste, com baixa inovação ou foco apenas no ciclo atual do software.

Adicionalmente, há fatores combinados que contribuem para a abordagem de teste: “prazo e orçamento”, “conhecimento técnico da equipe e método de desenvolvimento” e “tipo de sistema”. Prazo e orçamento são temas relacionados exclusivamente a uma etapa específica do ciclo de vida do projeto, reduzindo a possibilidades de implementar automatizações e uso de novas técnicas para serem utilizadas no futuro. Tais implementações são potenciais redutores de custos e tempo em manutenções ou implementações de novas funcionalidades. O conhecimento técnico e experiência dos gestores tendem a executar as atividades de forma padrão. Mesmo em equipes verticalizadas com métodos “ágeis”, notou-se ausência de integração e inovação. O tipo de sistema é a categoria que mais força o uso de ferramentas de teste, principalmente para validar os requisitos não funcionais.

5.6 Ameaças à validade

Levando em conta que se trata de um projeto de pesquisa, vale lembrar que ele é influenciado por alguns fatores que impõem ameaças à validade interna, à validade externa e à validade de conclusão.

A validade interna verifica fatores que podem ter afetado a pesquisa. Neste trabalho, por tratar de entrevistas, os entrevistados podem ter respondido de uma maneira subjetiva e não da forma que realmente executam as atividades de teste. Mesmo com o aspecto de confidencialidade garantido, poderiam existir deficiências ou ausências de controles na condução dos trabalhos que os entrevistados deixaram de citar para não prejudicar a imagem da empresa ou sua falta de capacitação para a gestão das atividades.

A validade externa define as condições que limitam a habilidade de generalizar os resultados. Os resultados obtidos sobre como é realizada a atividade de teste em ambientes industriais de desenvolvimento foi limitada ao grupo de empresas que não representa a totalidade dos modelos empresariais existentes, nem todos os segmentos de negócios e também as possíveis metodologias de desenvolvimento utilizadas na indústria de software. Porém, a base para a teoria apresentada representa uma parte significativa da indústria, tanto em modelo organizacional, porte e segmento. Deste universo, o processo de desenvolvimento utilizando os serviços de fábrica de software não foi avaliado e, por consequência, poderia surgir informações adicionais ao trabalho.

A validade de conclusão avalia a legitimidade do correlacionamento das variáveis que podem induzir a um resultado equivocado. A pesquisa qualitativa com a técnica TF atingiu a saturação teórica nas investigações considerando a amostragem das empresas e profissionais entrevistados, representando de forma adequada as conclusões apresentadas.

5.7 Considerações finais

Neste capítulo foi apresentada as características das empresas que participaram do experimento, os profissionais entrevistados e como foi conduzida as atividades de análise dos resultados.

Como resultado da aplicação da técnica TF, conclui-se que a estrutura organizacional direciona a adoção de modelos e técnicas de teste, com pouca inovação ou foco apenas no ciclo atual do software. Fatores adicionais, como prazo e orçamento, reduzem a possibilidade de automatização. A experiência dos gestores pode contribuir para inovação. O tipo de sistema é a categoria que mais direciona na escolha da técnica de teste.

No próximo capítulo, serão apresentados as conclusões e trabalhos futuros.

6 Conclusões

Este capítulo sintetiza os pontos importantes da condução do experimento realizado com gestores em cinco empresas que desenvolvem software. O foco da investigação foi em entender como são realizadas as atividades de teste na indústria de software e desenvolver uma teoria sobre a organização da atividade de teste.

6.1 Síntese do trabalho

As atividades de teste podem variar intensamente dependendo de fatores como orçamento de teste, método de desenvolvimento utilizado, perfil do cliente, tipo de software, entre outros. O estudo avaliou as atividades de teste em cinco empresas brasileiras desenvolvedoras de software.

Por meio de uma pesquisa qualitativa utilizando a teoria fundamentada (TF) foram realizadas coleta de dados com cinco profissionais da área de teste. Foi elaborado um guia de entrevista semiestruturada para ser o instrumento de coleta em campo. Ele foi utilizado durante a coleta de dados com cinco gestores de teste participantes. Em seguida, na etapa de análise, as entrevistas foram transcritas e inseridas no software [ATLAS.ti Scientific Software Development GmbH \(2017\)](#). O Atlas.ti é uma ferramenta que apoia a análise qualitativa, em especial a aplicação da TF; ele possui funcionalidades para organizar e gerir o material de forma sistemática, adequada para aplicar a TF. O objetivo da TF é formar uma teoria a partir do desenvolvimento e relacionamentos das categorias interpretadas nos dados analisados.

Como resultado final da análise, o estudo observou quatro pontos relevantes sobre como as atividades de teste são organizadas nas empresas que participaram desta pesquisa:

1. *Dependência do modelo organizacional.* A formação da equipe em modelo terceirizado, na qual prestadores de serviço participam do desenvolvimento, direciona a adoção de modelos e técnicas básicas de teste, com baixa inovação ou foco apenas no ciclo atual do software.
2. *Prazo e orçamento.* Foco apenas na fase atual do projeto reduz a possibilidades de implementar automatizações e uso de novas técnicas para serem utilizadas no futuro (e.g., manutenções, novas funcionalidades).

3. *Conhecimento técnico e experiência.* A escolha de técnicas e inovação é dependente do gestor e da integração da equipe.
4. *Tipo de sistema.* Categoria que mais induz ao uso de ferramentas de teste, principalmente para validar os requisitos não funcionais.

6.2 Contribuições

As principais contribuições apresentadas por esse trabalho são descritas a seguir.

1. *Uso de teoria fundamentada em atividades de teste no Brasil.* Que seja de nosso conhecimento, essa é a primeira pesquisa qualitativa utilizando teoria fundamentada para caracterizar a atividade de teste em empresas brasileiras de desenvolvimento de software.
2. *Teoria da atividade de teste em empresas brasileiras.* Foi desenvolvido um modelo da atividade de teste a partir do uso da técnica teoria fundamentada em cinco empresas brasileiras.
3. *Identificação de fatores relevantes para a atividade de teste.* Observou-se que a estrutura organizacional direciona a escolha das técnicas e ferramentas, bem como o tipo do sistema; o prazo e orçamento condicionam a utilização de técnicas de automatização.

6.3 Trabalhos futuros

Este trabalho é o início de um esforço de pesquisa qualitativa que visa caracterizar a atividade de teste na prática. Ele baseou-se apenas nas informações dos gestores de teste. Porém, ele pode ser estendido por meio da coleta de dados dos desenvolvedores. Por exemplo, eles podem ser observados *in loco* realizando as atividades de teste; entrevistas e questionários específicos para esses atores podem ser utilizados. Dessa maneira, será possível caracterizar a atividade de teste de outro ponto de vista.

Além disso, estudos qualitativos semelhantes podem ser realizados considerando empresas desenvolvedoras de software específicos (e.g., sistemas embarcados), empresas nas quais a tecnologia é a atividade fim, empresas nas quais a tecnologia é atividade meio e empresas com modelo de negócio baseado em fábrica de software.

Referências¹

AMMANN, P.; OFFUTT, J. *Introduction to Software Testing*. 2. ed. New York, NY, USA: Cambridge University Press, 2017. Citado 3 vezes nas páginas 18, 21 e 22.

ATLAS.ti Scientific Software Development GmbH. *Atlas.ti*. 2017. Disponível em: <http://atlasti.com>. Citado 3 vezes nas páginas 50, 53 e 63.

BECK, K. *Test Driven Development: By Example*. [S.l.]: Addison-Wesley Professional, 2002. Citado 2 vezes nas páginas 22 e 34.

BECK, K.; ANDRES, C. *Extreme Programming Explained: Embrace Change (2Nd Edition)*. [S.l.]: Addison-Wesley Professional, 2004. ISBN 0321278658. Citado na página 22.

BERLOWSKI, J. et al. Highly automated agile testing process: An industrial case study. *e-Informatica Software Engineering Journal*, v. 10, n. 1, p. 69–87, 2016. Citado 2 vezes nas páginas 34 e 37.

COLEMAN, G.; O'CONNOR, R. Investigating software process in practice: A grounded theory perspective. *Journal of Systems and Software*, v. 81, n. 5, p. 772–784, 2008. Citado 2 vezes nas páginas 34 e 38.

CRUZES, D. et al. How is security testing done in agile teams? a cross-case analysis of four software teams. *Lecture Notes in Business Information Processing*, v. 283, p. 201–216, 2017. Citado 2 vezes nas páginas 34 e 40.

DEAK, A.; STALHANE, T. Organization of testing activities in norwegian software companies. In: *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*. [S.l.: s.n.], 2013. p. 102–107. Citado 2 vezes nas páginas 34 e 35.

DELAMARO M. E.; MALDONADO, J. C. M. *Introdução ao teste de software*. [S.l.]: Elsevier, 2007. Citado 2 vezes nas páginas 15 e 24.

DELAMARO M. E.; CHAIM, M. L. A. M. R. *Técnicas e Ferramentas de Teste de Software*. 1. ed. [S.l.]: BOOKMAN, 2010. Citado 2 vezes nas páginas 30 e 31.

GREILER, M.; DEURSEN, A. van; STOREY, M. A. Test confessions: A study of testing practices for plug-in systems. In: *2012 34th International Conference on Software Engineering (ICSE)*. [S.l.: s.n.], 2012. p. 244–254. ISSN 0270-5257. Citado 2 vezes nas páginas 34 e 37.

HENDRICK, D. *Using Grounded Theory to Develop a Framework for Software Testing Best Practice in a Telecommunications Company*. Tese (Doutorado) — Dublin Institute of Technology, 2013. Citado 2 vezes nas páginas 34 e 39.

IEEE. *IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990)*. 1990. Disponível em: http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf. Citado na página 18.

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

- JUNIT. *JUnit 5 Testing Framework*. 2018. [Http://www.junit.org/](http://www.junit.org/). Acessado em 02 de fevereiro de 2018. Citado na página 31.
- MATHUR, A. P. *Foundations of software testing*. 2. ed. [S.l.]: Pearson, 2009. Citado 2 vezes nas páginas 15 e 23.
- MCCABE, T. A complexity measure. v. 2, n. 4, p. 308–320, dez. 1976. Citado na página 27.
- MCGREGOR, J. D. Test early, test often. v. 6, n. 4, p. 7–14, maio 2007. Available on-line at: http://www.jot.fm/issues/issue_2007_05/column1 [08-08-2007]. Citado na página 22.
- MOCKUS, A.; NAGAPPAN, N.; DINH-TRONG, T. T. Test coverage and post-verification defects: A multiple case study. In: *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. [S.l.: s.n.], 2009. p. 291–301. ISSN 1949-3770. Citado 2 vezes nas páginas 34 e 36.
- MOIR, K. Releng of the nerds: Open source release engineering. SDK code coverage with JaCoCo. 2011. Disponível em: <http://relengofthenerds.blogspot.com.br/2011/03/sdk-code-coverage-with-jacoco.html>. Citado na página 29.
- MYERS G. J.; SANDLER, C. T. T. *The Art of Software Testing*. [S.l.]: Wiley, New York, 2004. Citado 3 vezes nas páginas 15, 24 e 25.
- PRESSMAN, R. S. *Engenharia de Software*. 6. ed. [S.l.]: McGraw-Hill, 2006. Citado 4 vezes nas páginas 22, 24, 30 e 31.
- ROBSON, C. *Real World Research*. 3. ed. [S.l.]: Wiley, 2011. Citado 3 vezes nas páginas 43, 44 e 45.
- SELENIUM. *Selenium IDE – Integrated development environment for Selenium tests*. 2018. [Http://www.seleniumhq.org/projects/ide/](http://www.seleniumhq.org/projects/ide/). Acessado em 01 de fevereiro de 2018. Citado na página 31.
- SOMMERVILLE, I. *Engenharia de Software*. 9. ed. [S.l.]: Addison-Wesley Brasil, 2011. Citado 2 vezes nas páginas 21 e 23.
- STRAUSS A.; CORBIN, J. *Pesquisa Qualitativa - Técnicas e procedimentos para o desenvolvimento de pesquisa fundamentada*. 2. ed. [S.l.]: BOOKMAN, 2008. Citado 4 vezes nas páginas 17, 42, 45 e 46.
- TAIPALE, O.; SMOLANDER, K. Improving software testing by observing practice. In: *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*. New York, NY, USA: ACM, 2006. (ISESE '06), p. 262–271. ISBN 1-59593-218-6. Citado 2 vezes nas páginas 34 e 40.
- THOMSON, C. D.; HOLCOMBE, M.; SIMONS, A. J. H. What makes testing work: Nine case studies of software development teams. In: *2009 Testing: Academic and Industrial Conference - Practice and Research Techniques*. [S.l.: s.n.], 2009. p. 167–175. Citado na página 34.
- VINCENZI, A. M. R. et al. *Automatização de Teste de Software com Ferramentas de Software Livre*. Primeira. Rio de Janeiro: Elsevier, 2018. No prelo. Citado na página 21.

ZELLER, A. *Why Programs Fail: A Guide to Systematic Debugging*. [S.l.]: Morgan Kaufmann Publishers Inc, 2005. Citado 2 vezes nas páginas 18 e 20.

Apêndice A – Utilização da ferramenta JaCoCo

A EclEmma/JaCoCo coleta as informações de cobertura de fluxo de controle por meio da execução do programa. Há várias formas de executar o programa e obter a cobertura de código, sendo as mais comuns por meio da execução do programa como uma aplicação Java local ou a execução de casos de teste automatizados por meio de bibliotecas como *JUnit*¹, *TestNG*² e *SWTBot*³. Como EclEmma/JaCoCo instrumenta o código compilado em Bytecodes, a ferramenta também pode coletar a cobertura de aplicações escritas em Scala⁴.

O exemplo de utilização da EclEmma/JaCoCo obtém a cobertura por meio de uma chamada ao método `CalcularMDC()` utilizando a biblioteca *JUnit*. A classe com a chamada do teste está descrita na Figura 9.

Figura 9 – Classe de teste do método `calcularMDC`

```

1 package Programa.Exemplo;
2
3 import static org.junit.Assert.*;
4 import org.junit.Test;
5
6 public% class MDCTest {
7
8     Public void testCalcularMDC() {
9
10         assertEquals(8, MDC.calcularMDC(8, 16));
11
12     }
13 }
```

Fonte: Sérgio Luis Barbieri, 2018

Por padrão, EclEmma/JaCoCo executa todos os métodos com a anotação `@Test` da biblioteca *JUnit*. Ao clicar no botão de cobertura o teste é executado e a cobertura coletada para todo o código. É possível determinar as classes que serão avaliadas com relação à análise de cobertura. Para tanto, deve-se utilizar o submenu disponível no botão de cobertura e selecionar a opção *Coverage Configurations...* Outra alternativa é executar

---

¹ <http://junit.org/junit4/>
² <http://testng.org/doc/index.html>
³ <http://www.eclipse.org/swtbot/>
⁴ <https://www.scala-lang.org/>

o teste no menu *Run as JUnit Test* e após a finalização dos testes, gerar a cobertura no menu *Run Coverage Last Launched*.

A.1 Relatórios Gerados e Análise dos Dados Produzidos

Ao término da cobertura o Eclipse abre automaticamente a perspectiva *Coverage*. A Figura 10 apresenta uma visão geral sobre os testes executados. No exemplo, são apresentados os resultados para o método `calcularMDC()`. Note-se que com a execução do teste (cálculo do máximo divisor comum entre 8 e 16) foi obtida a cobertura de 16,7% considerando o critério complexidade (*Complexity*), ou seja, o método possui cinco caminhos e o teste percorreu apenas um caminho.

Figura 10 – Cobertura de complexidade utilizando a EclEmma/JaCoCo

Element	Coverage	Covered Complexity	Missed Complexity	Total Complexity
MDC	30,0 %	3	7	10
src	30,0 %	3	7	10
Livro.Exemplo	30,0 %	3	7	10
MDC.java	12,5 %	1	7	8
MDC	12,5 %	1	7	8
calcularMDC(int, int)	16,7 %	1	5	6
main(String[])	0,0 %	0	1	1
MDCTest.java	100,0 %	2	0	2

Fonte: Sérgio Luis Barbieri, 2018

Trocando a visão de cobertura para instruções (*Instructions*) utilizando o menu *View* (Figura 11), verifica-se que o método `calcularMDC()` possui 43 instruções e o teste teve uma cobertura de 28 instruções. Observe-se que o número de instruções é maior que o número de linhas do Programa 14. Isso ocorre porque a EclEmma/JaCoCo instrumenta o código compilado em Bytecodes que, por sua vez, possui mais instruções. A Figura 12 mostra a visão de ramos (*Branches*) que apresenta cobertura de 50%, ou seja, metade dos ramos foram cobertos pelo caso de teste.

A Figura 13 apresenta o resultado em forma de resumo da análise de cobertura, chamada de *Session* pela EclEmma/JaCoCo. A janela é apresentada utilizando a opção *Properties* a partir de qualquer “Elemento” da perspectiva *Coverage*. Note-se que além das coberturas discutidas nas Figuras 10, 11 e 12 o resumo também apresenta a cobertura

Figura 11 – Cobertura de instruções utilizando a EclEmma/JaCoCo

Element	Coverage	Covered Instruções...	Missed Instructions	Total Instructions
MDC	52,8 %	38	34	72
src	52,8 %	38	34	72
Livro.Exemplo	52,8 %	38	34	72
MDC.java	45,2 %	28	34	62
MDC	45,2 %	28	34	62
calcularMDC(int, int)	65,1 %	28	15	43
main(String[])	0,0 %	0	16	16
MDCTest.java	100,0 %	10	0	10

Fonte: Sérgio Luis Barbieri, 2018

Figura 12 – Cobertura de ramos utilizando EclEmma/JaCoCo

Element	Coverage	Covered Branches	Missed Branches	Total Branches
MDC	50,0 %	5	5	10
src	50,0 %	5	5	10
Livro.Exemplo	50,0 %	5	5	10
MDC.java	50,0 %	5	5	10
MDC	50,0 %	5	5	10
calcularMDC(int, int)	50,0 %	5	5	10
main(String[])	0,0 %	0	0	0
MDCTest.java	0,0 %	0	0	0

Fonte: Sérgio Luis Barbieri, 2018

de linhas, métodos e tipos (classes). As informações de cobertura da Figura 13 envolvem todo o programa e não apenas o método `calcularMDC()`.

A Figura 14 apresenta a cobertura no código para a execução do método `calcularMDC()` com os valores 8 e 16 para os parâmetros `valor1` e `valor2`, respectivamente. Para todos os métodos executados o sistema registra qual foi a cobertura para cada linha de código: linhas em vermelho — não exercitadas; linhas em verde — totalmente exercitadas; e linha em amarelo — parcialmente exercitada. As linhas amarelas estão associadas a comandos de decisão. Uma linha amarela nesses casos significa que apenas uma das condições associadas ao comando foi exercitada pelos casos de teste.

Adicionalmente, ao lado esquerdo da linha, é apresentado um ícone diamante com informações adicionais em relação à cobertura. Ao passar o *mouse* sobre o ícone da linha 11 será apresentado o número de desvios não cobertos.

Figura 13 – Resumo da análise de cobertura realizada pela EcEmma/JaCoCo

Fonte: Sérgio Luis Barbieri, 2018

Para obter-se 100% da cobertura dos desvios é necessário o conjunto de testes apresentado na Figura 15:

Figura 14 – Análise de cobertura realizada pela EcEmma/JaCoCo no código fonte

Fonte: Sérgio Luis Barbieri, 2018

Figura 15 – Trecho de código para cobertura de 100% do método calcularMDC

```

1     Public void testCalcularMDC() {
2
3         assertEquals(8, MDC.calcularMDC(8, 16));
4         assertEquals(1, MDC.calcularMDC(7, 4));
5         assertEquals(10, MDC.calcularMDC(10, 10));
6         assertEquals(-1, MDC.calcularMDC(0, 2));
7         assertEquals(-1, MDC.calcularMDC(2, 0));
8
9     }
10 }

```

Fonte: Sérgio Luis Barbieri, 2018

Apêndice B – Termo de consentimento livre e esclarecido

A seguir é apresentado o modelo original do termo submetido aos participantes das entrevistas.

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Título do projeto de pesquisa: Teste de software na indústria: Um estudo qualitativo.

Pesquisador: Sérgio Luis Barbieri.

O (a) Sr. (a) está sendo convidado (a) a participar da pesquisa “Teste de software na indústria: Um estudo de caso qualitativo”. Nesta pesquisa pretendemos identificar as técnicas de teste utilizadas pelas equipes de desenvolvimento, práticas adotadas, quais métricas e critérios são definidos nos processos de aceite e quais ferramentas são adotadas. O objetivo da pesquisa é entender como a indústria aplica as técnicas de teste e quais ferramentas de teste são adotadas no ciclo de desenvolvimento de sistemas.

O (a) Sr. (a) tem o direito de interromper sua participação na pesquisa a qualquer momento. Os riscos associados à sua participação são mínimos.

A participação compreende em uma entrevista que terá a duração de 30 minutos direcionada aos gestores da área, a aplicação do questionário de 15 minutos para os profissionais envolvidos no processo de teste e o acompanhamento de uma sessão de teste como observador.

Leia atentamente os itens a seguir e assine no campo adequado deste termo caso aprove as seguintes condições:

1. A entrevista será gravada e uma transcrição será realizada.
2. Você receberá a transcrição da entrevista e poderá corrigir seu conteúdo caso haja erros na transcrição.
3. A transcrição da entrevista será realizada e analisada pelo pesquisador principal do projeto: Sérgio Luis Barbieri.
4. O acesso à transcrição da entrevista será limitado ao pesquisador principal do projeto, a acadêmicos e demais colaboradores desta pesquisa.
5. As sessões de teste serão observadas e transcritas pelo pesquisador principal do projeto: Sérgio Luis Barbieri.
6. O acesso à transcrição das sessões de teste observadas será limitado ao pesquisador principal do projeto, a acadêmicos e demais colaboradores desta pesquisa.
7. Resumos do conteúdo das entrevistas e sessões de teste ou citações diretas a trechos delas, utilizados em publicações acadêmicas ou outros meios acadêmicos serão apresentados de forma anônima de maneira a garantir que sua identidade não seja revelada. Além disso, cautelas serão tomadas para que qualquer outra informação da entrevista ou da sessão de teste que possa levar sua identificação seja omitida.

8. A gravação da entrevista será mantida para registro da pesquisa e possível análise por outros pesquisadores durante um período de cinco anos. Durante esse período, todos os cuidados de sigilo serão mantidos quanto à utilização do conteúdo da entrevista.

9. O questionário não necessitará da identificação do respondente.

10. Qualquer alteração dos itens acima ocorrerá apenas com a sua aprovação.

Acordo de citação

No que diz respeito a ser citado, favor assinalar ao lado de qualquer uma das declarações com as quais concorda.

- ☐ Em relação a citação, entendo que minhas palavras podem ser citadas diretamente.
- ☐ Desejo rever as notas, transcrições ou outros dados relacionados à minha participação, recolhidos durante a entrevista ou sessão de teste.
- ☐ Concordo em ser citado diretamente.
- ☐ Concordo em ser citado diretamente se meu nome não for publicado e um pseudônimo for utilizado.
- ☐ Concordo que os pesquisadores podem publicar documentos que contenham citações minhas.

Todo ou parte do conteúdo da entrevista poderá ser usado:

- 1. Em eventos acadêmicos.
- 2. Em artigos acadêmicos e apresentações de trabalhos.
- 3. Em nosso *website* e outros meios de divulgação como apresentações e eventos de discussão científica.
- 4. Em um arquivo do projeto conforme observado acima.

Ao assinar este formulário eu concordo que:

- 1. Participo voluntariamente deste projeto.
- 2. Eu entendo que posso interromper minha participação na entrevista ou da sessão de teste a qualquer momento.
- 3. A entrevista ou sessão de teste transcrita ou os seus fragmentos podem ser utilizados como descrito acima.
- 4. Não espero receber qualquer benefício ou pagamento por minha participação.

5. Posso solicitar uma cópia da transcrição da minha entrevista ou da sessão de teste que participei, bem como realizar alterações para assegurar a eficácia o acordo de confidencialidade.

6. Posso fazer perguntas antes, durante e depois da entrevista ou sessão de teste.

7. Tive oportunidade de perguntar qualquer dúvida que tenha surgido e que sou livre para entrar em contato com o pesquisador a qualquer momento, sanando possíveis dúvidas em relação a entrevista que possam surgir no futuro.

Os procedimentos éticos para pesquisas acadêmicas exigem que os entrevistados concordem explicitamente em serem entrevistados e como as informações contidas em sua entrevista serão usadas

_____	____/____/____
Assinatura do participante	Data
_____	____/____/____
Assinatura do pesquisador	Data

Informações de contato

Em caso de dúvidas sobre este estudo, por favor entre em contato:

Nome do pesquisador: Sérgio Luis Barbieri.

Endereço: Rua Professor Pedreira de Freitas, 179 AP 131. Bairro: Tatuapé, São Paulo - SP, CEP: 03312-052

Tel.: (11) 94071-8994

E-mail: sergio.barbieri@usp.br.

Este termo de consentimento encontra-se impresso em duas vias originais, sendo que uma será arquivada pelo pesquisador responsável, na Universidade de São Paulo e a outra será fornecida ao (a) Sr. (a). Os dados e instrumentos utilizados na pesquisa ficarão arquivados com o pesquisador responsável por um período de 5 (cinco) anos, e após esse tempo serão destruídos. Em caso de dúvidas e problemas o (a) Sr. (a) poderá se dirigir diretamente ao pesquisador principal desse projeto.