UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

ISSN 0103-2569

RELAÇÃO ENTRE ARQUITETURA DE SOFTWARE E TESTE DE SOFTWARE: UM MAPEAMENTO SISTEMÁTICO

NILTON MENDES DE SOUZA
DIÓGENES DIAS SIMÃO
LUCAS BUENO RUAS DE OLIVEIRA
CRISTIANE APARECIDA LANA
ELISA YUMI NAKAGAWA
JOSÉ CARLOS MALDONADO

Nº 415

RELATÓRIOS TÉCNICOS

São Carlos – SP
Fev./2017

Resumo

O projeto de arquitetura de software e teste de software são duas atividades essenciais para o desenvolvimento de produtos de software de alta qualidade. Considerando ser fundamental a representação de informações/requisitos de teste de software nos estágios iniciais do desenvolvimento de software, assim como o uso de todos os artefatos de software para a determinação de requisitos de teste, este relatório técnico visa a caracterizar o estado da arte de como as atividades de projeto de arquitetura de software e teste de software têm sido exploradas de forma complementar no processo de desenvolvimento de software, por meio da condução de um mapeamento sistemático. Foram identificados 27 estudos que exploram o uso de informações relacionadas à arquitetura de software para apoiar a atividade de teste de software. As abordagens utilizadas abrangem as fases de teste de integração, unidade e regressão, por meio das técnicas estruturais e funcionais. Com este estudo, pode-se observar que são poucas as iniciativas que exploram conjuntamente as atividades de projeto de arquitetura de software e teste de software, principalmente no que tange a representar informações/requisitos de teste de software no mesmo nível de abstração de arquiteturas de software.

Sumário

1 Introdução **1**

2 Mapeamento Sistemático: Relação entre Arquitetura de Software e Teste de Software **5**
2.1 Considerações Iniciais 5
2.2 Conceitos e Definições Básicos 6
2.3 Mapeamento Sistemático: Fase 1 – Planejamento 8
2.3.1 Objetivo e Questões de Pesquisa 8
2.3.2 Critérios de Seleção 8
2.3.3 Fontes de Seleção 9
2.3.4 Seleção dos Estudos 10
2.4 Mapeamento Sistemático: Fase 2 - Condução 10
2.4.1 Primeira Etapa 10
2.4.2 Segunda Etapa 11
2.5 Mapeamento Sistemático: Fase 3 - Apresentação dos Resultados 12
2.5.1 RQ-1: Quais abordagens têm explorado arquitetura de software e teste de software em conjunto no processo de desenvolvimento de software? 12
2.5.2 RQ-2: Quais informações arquiteturais têm sido consideradas para apoiar a atividade de teste? 14
2.5.3 RQ-3: Existem ferramentas que oferecem suporte às abordagens identificadas? 16
2.6 Considerações Finais 18

3 Visão Geral da Área de Pesquisa **19**
3.1 Considerações Iniciais 19
3.2 Quantidade de Estudos Publicados por Ano 19
3.3 Tipos de Avaliações 20
3.4 Autores por País 21
3.5 Técnicas e Fases de Teste 21
3.6 Relacionamento entre Autores 23
3.7 Relacionamento entre Veículos de Publicação e Avaliações 23
3.8 *Wordcloud* 24
3.9 Ameaças à Validade 25

3.10 Considerações Finais 26

4 Conclusões e Perspectivas **27**

A Autores Identificados **35**

B Strings de busca **39**
B.1 Strings adaptadas às bases 40
B.1.1 Strings aplicadas à IEEE 40
B.1.2 Strings aplicadas à ACM 42
B.1.3 String aplicadas à Science Direct 43
B.1.4 String aplicadas à Web of Science 44
B.1.5 String aplicada à SCOPUS 44

C Formulário para extração de dados **45**

D Veículos de Publicação Identificados **47**

Lista de Figuras

3.1 Quantidade de estudos por ano 20
3.2 Quantidade de autores distribuídos por país e por ano 21
3.3 Fases de teste e técnicas de teste 22
3.4 Grafo da relação entre autores e estudos publicados. 23
3.5 Tipos de veículo de publicação e tipo avaliações conduzidas 24
3.6 *Wordcloud* das palavras contidas nos resumos dos estudos 25

B.1 Adaptação da string para IEEE: “Abstract” x “Abstract”) 40
B.2 Adaptação da string para IEEE: “Author Keywords” x “Abstract”) 40
B.3 Adaptação da string para IEEE: “Abstract” x “‘Author Keywords”) 40
B.4 Adaptação da string para IEEE: “Abstract” x “‘Document Title”) 40
B.5 Adaptação da string para IEEE: “Document Title” x “‘Abstract”) 41
B.6 Adaptação da string para IEEE: “Author Keywords” x “‘Author Keywords”) 41
B.7 Adaptação da string para IEEE: “Document Title” x “‘Document Title”) . 41
B.8 Adaptação da string para IEEE: “Document Title” x “‘Author Keywords”) 41
B.9 Adaptação da string para IEEE: “Author Keywords” x “‘Document Title”) 41
B.10 Adaptação da string para ACM: “Abstract” x “Abstract”) 42
B.11 Adaptação da string para ACM: “Abstract” x “Keywords”) 42
B.12 Adaptação da string para ACM: “Keywords” x “Abstract”) 42
B.13 Adaptação da string para ACM: “Abstract” x “Title”) 42
B.14 Adaptação da string para ACM: “Title” x “Abstract”) 42
B.15 Adaptação da string para ACM: “Keywords” x “Keywords”) 43
B.16 Adaptação da string para ACM: “Title” x “Title”) 43
B.17 Adaptação da string para ACM: “Title” x “Keywords”) 43
B.18 Adaptação da string para ACM: “Keywords” x “Title”) 43
B.19 Adaptação da string para Science Direct. 43
B.20 Adaptação da string para Web of Science. 44
B.21 Adaptação da string para SCOPUS. 44

C.1 Formulário para extração de dados 45

Capítulo 1

Introdução

Nos últimos anos, é notável o aumento do tamanho e da complexidade dos sistemas de software. Nesse sentido, a arquitetura de software tem desempenhado um papel importante para propiciar um melhor entendimento desses sistemas [16]. Por outro lado, as atividades de VV&T (Verificação, Validação e Teste) são ainda mais relevantes, pois geralmente esses sistemas usualmente envolvem riscos econômicos e sociais. Assim sendo, o projeto de arquitetura de software e teste de software são duas atividades essenciais para o desenvolvimento de produtos de software de alta qualidade. Essas atividades têm sido amplamente exploradas na literatura nas últimas décadas.

Por meio da arquitetura de software, é iniciado o planejamento de sistemas e de seus atributos de qualidade, contribuindo diretamente para o controle da viabilidade e da qualidade do produto a ser desenvolvido [2]. A arquitetura de software, como disciplina, começou a receber atenção no início da década de 90 e, desde então, vem sendo explorada na academia e na indústria [12]. Arquiteturas de software desempenham um papel fundamental para prover qualidade no desenvolvimento de sistemas de software grandes e complexos [36]. Ao projetar uma arquitetura de software, busca-se satisfazer os requisitos do sistema e cabe ao arquiteto de software, durante o projeto arquitetural, decidir a maneira mais adequada para atender esses requisitos [17].

O teste de software é reconhecido como uma das atividades indispensáveis no desenvolvimento de sistemas de software [10]. O teste de software está inserido em um conjunto de atividades denominadas VV&T e tem por objetivo revelar falhas existentes no programa em teste e identificar as inconformidades em relação à especificação do sistema [34]. Assim como para o processo de desenvolvimento de software, foram definidas iniciativas para caracterizar modelos de maturidade do processo de teste [1], a exemplo do TMMi (do inglês, *Test Maturity Model Integration*) [47].

Segundo Harrold [15] e Myers [34], aplicando-se as atividades de VV&T nos estágios iniciais do desenvolvimento, nos quais é incluído o projeto de arquitetura de software,

contribui-se para que o processo de desenvolvimento de software seja mais efetivo. Assim, considera-se ser fundamental a representação de informações/requisitos de teste de software nos estágios iniciais do desenvolvimento de software, bem como o uso de todos os artefatos de software para a determinação de requisitos de teste.

Dessa forma, o objetivo deste estudo é caracterizar o estado da arte de como as atividades de projeto de arquitetura de software e teste de software têm sido exploradas de forma complementar no processo de desenvolvimento de software. Para fornecer esse panorama, foi conduzido um mapeamento sistemático [38], uma técnica que auxilia a encontrar e sumarizar evidências disponíveis sobre um tópico de pesquisa específico. O mapeamento sistemático tem sido utilizado para compreender, sumarizar e prover uma visão geral do estado da arte em um tópico de pesquisa. De acordo com Petersen et al. [38], por meio de mapeamentos sistemáticos a literatura é revisitada para compreender como tópicos de pesquisa têm sido explorados e em quais veículos eles têm sido publicados. Esse processo é composto pelas seguintes fases: planejamento, condução e apresentação dos resultados. A condução do mapeamento sistemático apresentado neste documento foi baseado no processo proposto por Kitchenham e Charters [23] em conjunto com a técnica *backward snowballing* [49] e na opinião de especialistas. Além disso, também foram consideradas as diretrizes de classificação de resultados propostas por Petersen et al. [38].

Os estudos incluídos nesse mapeamento sistemático tratam essencialmente o teste baseado em arquitetura, que consiste em usar a arquitetura como um artefato para derivar requisitos/casos de teste. Na maioria dos estudos foi reportado o teste de conformidade entre a arquitetura e o sistema implementado, principalmente considerando o teste funcional na fase de integração. As fases de teste de unidade e regressão também foram mencionadas; porém, em apenas cinco e sete estudos, respectivamente. A fase de teste de sistema não foi mencionada nos estudos. Considerando as técnicas de teste, somente a funcional e a estrutural foram utilizadas. Além disso, foram consideradas como fontes de informações arquiteturais: análise estática, análise dinâmica, transformações de modelos arquiteturais e gerenciamento de configurações da arquitetura.

Quanto à automatização da execução da atividade de teste, apenas em cinco estudos foram utilizadas ferramentas ou ambientes. Somente 17 dos 27 estudos (63%) tiveram alguma avaliação, predominantemente por meio estudos de caso. Praticamente os estudos foram realizados no ambiente acadêmico, com exceção de dois deles realizados na indústria: um estudo de caso e um experimento.

A estrutura deste relatório está organizada da seguinte forma. No Capítulo 2, são apresentados o planejamento, a condução e a apresentação dos resultados do mapeamento sistemático. No Capítulo 3, é apresentada uma visão geral da área de pesquisa

considerando o relacionamento entre arquitetura de software e teste de software. Por fim, no Capítulo 4, apresentam-se as conclusões e as propostas para trabalhos futuros.

Capítulo 2

Mapeamento Sistemático: Relação entre Arquitetura de Software e Teste de Software

2.1 Considerações Iniciais

No início dos anos 2000, Harrold [15] apresentou que o teste de software poderia ser beneficiado com artefatos produzidos antes da implementação ("*precode artifacts*"), como requisitos e arquitetura de software. Além disso, a autora aponta que a especificação arquitetural pode ser utilizada, por exemplo, para verificar a testabilidade de um sistema, o teste de unidade, o teste de integração e o teste de regressão. Bertolino et al. [5] relacionam arquitetura de software e o teste de software em três tópicos de pesquisa, a saber: avaliação arquitetural, análise baseada em arquitetura e teste baseado em arquitetura. Avaliação arquitetural consiste em avaliar a conformidade entre arquitetura de software e os requisitos de qualidade. Na análise baseada em arquitetura, analisam-se arquiteturas ou modelos arquiteturais alternativos em relação às qualidades funcionais e não funcionais esperadas do sistema, de forma a subsidiar decisões de projeto nos estágios iniciais do desenvolvimento. Os artefatos arquiteturais são utilizados para selecionar a arquitetura que melhor satisfaz a qualidade esperada e os interesses dos *stakeholders*. Teste baseado em arquitetura consiste em utilizar artefatos arquiteturais para selecionar especificações de teste que serão utilizadas nos sistemas implementados a partir dessa arquitetura.

Neste capítulo, antes da apresentação do mapeamento sistemático conduzido e dos resultados obtidos, é pertinente sintetizar alguns conceitos básicos e definições sobre arquitetura de software e teste de software, realizado na Seção 2.2. O restante deste capítulo está organizado da seguinte forma. Na Seção 2.3, o planejamento do mapeamento sistemático é descrito. Na Seção 2.4, apresenta-se a condução do mapeamento sistemático. Na Seção 2.5, os resultados do mapeamento são descritos. Por fim, na Seção 2.6, as considerações finais sobre este capítulo são apresentadas.

2.2 Conceitos e Definições Básicos

• Arquitetura de Software

A arquitetura de software contém a descrição da estrutura do sistema, seus elementos e suas relações, que tem por objetivo expressar o comportamento do sistema [2]. Uma descrição arquitetural pode ser expressa por meio de visões arquiteturais [20]. Em geral, uma arquitetura é representada por meio de várias visões, pois apenas uma não contém informação suficiente para representar todos os interesses e pontos de vista sobre o sistema [9, 20]. Um estilo arquitetural define como serão dispostos os componentes e seus relacionamentos de uma arquitetura, bem como um conjunto de restrições [9, 19]. Exemplos de estilos arquiteturais são [9]: arquitetura orientada a serviço (do inglês, *Service Oriented Architecture* – SOA), cliente e servidor, arquitetura em camadas. As decisões tomadas durante o projeto da arquitetura são muito importantes para as fases subsequentes do desenvolvimento, implicando diretamente no código e na qualidade do sistema [11]. Heesch et al. [16] afirmam que a arquitetura de software também é reconhecida como um registro de decisões de projeto, tais como a escolha do estilo arquitetural e dos protocolos de comunicação entre os elementos do sistema.

Com o propósito de contribuir para o desenvolvimento de sistemas de software, a organização internacional OMG (do inglês, *Object Management Group*) propôs uma abordagem de arquitetura dirigida por modelo (do inglês, *Model Driven Architecture* – MDA) [45]. Essa abordagem possui modelos que devem ser representados em uma notação bem definida, sendo que a especificação do sistema deve ser organizada em um conjunto de modelos e suas transformações associadas [6]. Modelos de MDA podem ser representados em três níveis de abstração [6]: (i) Independente de Computação (do inglês, *Computation independent – CIM*), que se refere aos modelos de negócio ou domínio, além disso, representa o que o sistema deve fazer, mas sem apresentar qualquer detalhe técnico; (ii) Independente de Plataforma (do inglês, *Platform independent – PIM*), que apresenta um grau de abstração que pode ser implementado para uma ou mais plataformas; e (iii) Específico à Plataforma (do inglês, *Platform-specific – PSM*), que combina as especificações do PIM com detalhes requeridos para definir como um sistema usa uma plataforma em particular. A MDA permite a divisão de projetos em diferentes fases, o que impacta o ciclo de desenvolvimento do software [45].

• Teste de Software

O processo de desenvolvimento de sistemas de software não é uma tarefa trivial [10]. Segundo ISO/IEC/IEEE-29119-1 [18], não é possível desenvolver um sistema de software perfeito e, por isso, é necessário testá-lo antes que ele seja entregue aos usuários. O teste de software é uma atividade dinâmica que se baseia na execução de um programa ou modelo

com entradas específicas para analisar se a saída obtida da execução está de acordo com a saída esperada [10].

A atividade de teste envolve basicamente quatro etapas, planejamento de testes, projeto de casos de teste, execução e avaliação dos resultados dos testes, que devem ser desenvolvidas durante toda a atividade de teste [10]. Em geral, essa atividade é executada em fases com objetivos distintos, a saber [10, 34]:

- **Teste de unidade**: tem como objetivo testar unidades menores de um sistema. Essas unidades podem ser vistas como funções, métodos, classes ou sub-rotinas de um sistema. Nesse nível, o teste pode ser executado à medida que ocorre a implementação de uma unidade;

- **Teste de integração**: geralmente é realizado após a execução dos testes de unidade. Nesse nível, é dada ênfase na construção da estrutura do sistema, verificando se a interação entre as unidades ocorrem de maneira adequada;

- **Teste de sistema**: tem início quando as unidades do sistema e suas integrações foram devidamente testadas e estão funcionando corretamente. O objetivo desse nível é verificar se o sistema desenvolvido corresponde às especificações, abordando requisitos funcionais e não funcionais (tais como, segurança, performance e robustez); e

- **Teste de regressão**: durante a etapa de manutenção, diversas modificações são realizadas no sistema, assim, é possível que novos defeitos possam ter sido inseridos. Nesse sentido, torna-se importante realizar o teste de regressão para assegurar que as modificações implementadas no sistema não inseriram ou resultarem em defeitos em outras partes do sistema.

As principais técnicas que estabelecem diretrizes e critérios para a geração e avaliação de conjuntos de casos de teste (dado de teste e saída esperada) são [10, 34]:

- **Técnica Funcional:** também conhecida como teste de "caixa preta". Nessa técnica, a especificação do sistema é utilizada e a estrutura interna do programa não é considerada. Para que a aplicação do teste funcional seja efetiva, é importante que os requisitos estejam bem definidos, visto que essa técnica é baseada apenas na especificação do sistema. Particionamento em classes de equivalência, análise de valor limite e grafos causa-efeito são exemplos de critérios dessa técnica;

- **Técnica Estrutural:** também conhecida como "teste de caixa branca", baseia-se na implementação do sistema, ou seja, na estrutura interna do sistema a ser testado.

Nessa técnica, os critérios definem como os dados de teste deverão cobrir instruções, variáveis e laços. Assim, o usos de variáveis, laços de repetição e conjuntos específicos de condição são exercitados. Seus critérios, em geral, são classificados com base na complexidade dos sistemas, fluxo de controle e no fluxo de dados. Todos nós, todas arestas, todos os usos são exemplos de critérios dessa técnica;

- **Baseada em Defeitos:** Os critérios da técnica baseada em defeitos são caracterizados pela utilização dos defeitos recorrentes no processo de desenvolvimento de software para derivar requisitos de teste. O teste de mutação, também chamado de análise de mutantes, é o critério mais conhecido da técnica baseada em defeitos.

2.3 Mapeamento Sistemático: Fase 1 – Planejamento

Nessa fase, o protocolo de pesquisa do mapeamento sistemático é estabelecido e nele são definidos: (i) objetivo de pesquisa e questões de pesquisa; (ii) critérios de inclusão e exclusão; (iii) fontes de seleção; e (iv) seleção de estudos. O protocolo foi elaborado pelos autores e validado por especialistas em teste de software, arquitetura de software e mapeamento sistemático.

2.3.1 Objetivo e Questões de Pesquisa

O objetivo deste mapeamento sistemático é caracterizar o estado da arte de como as atividades de projeto de arquitetura de software e teste de software têm sido exploradas de forma complementar no processo de desenvolvimento de software. Para alcançar esse objetivo, foram definidas as seguintes questões de pesquisa:

RQ-1 Quais abordagens têm explorado arquitetura de software e teste de software em conjunto no processo de desenvolvimento de software?

RQ-2 Quais informações arquiteturais têm sido consideradas para apoiar a atividade de teste?

RQ-3 Existem ferramentas que ofereçam suporte às abordagens identificadas?

2.3.2 Critérios de Seleção

Para a seleção de estudos, foram definidos um Critério de Inclusão (CI) e cinco Critérios de Exclusão (CE), que são:

CI.1 - O estudo explora em conjunto arquitetura de software e teste de software;

CE.1 - O estudo não está escrito em inglês;

CE.2 - O estudo não relaciona arquitetura de software e teste de software;

CE.3 - No caso de estudos duplicados, o estudo mais completo é considerado para extração de dados;

CE.4 - O estudo é um índice de conferência, tutorial, *keynotes*; e

CE.5 - O estudo foi publicado na literatura cinzenta.

2.3.3 Fontes de Seleção

As fontes de seleção são os meios pelos quais os estudos serão obtidos para serem selecionados no mapeamento. Nesse mapeamento, foram consideradas três tipos de fontes de seleção, sendo elas máquinas de busca, *backward snowballing* e opinião de especialistas. As máquinas de busca foram selecionadas com base na opinião de especialistas e nas recomendações de Kitchenham e Charters [23]. Na Tabela 2.1 são apresentadas as máquinas de busca utilizadas.

Tabela 2.1: Máquinas de busca

Máquina de busca	***Web site***
ACM Digital Library	http://dl.acm.org/
IEEE Xplore	http://ieeexplore.ieee.org/
Science Direct	http://sciencedirect.com
Scopus	http://scopus.com/
Web of Science	http://webofscience.com/

Para utilizar as máquinas de busca, foi definida uma *string* de busca com os termos "testing" e "software architecture". A *string* de busca foi adequada a cada especifidade das máquinas de busca. Uma busca piloto utilizando apenas os termos `"testing" AND "software architecture"` foi conduzida e três estudos retornados ([3, 5, 33]) foram utilizados como estudos controle. Esses estudos foram selecionados por serem diretamente relacionados ao objetivo do mapeamento. A *string* de busca é apresentada na Tabela 2.2. Na primeira coluna são listadas as áreas da pesquisa e na segunda coluna os termos relacionados.

Tabela 2.2: *String* de busca

Área	**Termos**
Arquitetura de Software	("Software Architecture" OR "Software Architectures")
	AND
Teste de Software	(Testing OR "Test Activity" OR "Test Activities" OR "Test Plan" OR "Test Plans" OR "Test Criterion" OR "Test Criteria" OR "Test Case" OR "Test Cases")

2.3.4 Seleção dos Estudos

Os estudos primários retornados das máquinas de busca foram selecionados considerando o CI e os CE em duas etapas. Na primeira etapa, os critérios de seleção foram aplicados somente no título, resumo e palavras-chave. Na segunda etapa, os critérios de seleção foram aplicados após a leitura completa dos estudos. A técnica *backward snowballing* foi aplicada aos estudos incluídos na segunda etapa. Os critérios de seleção foram novamente aplicados nos estudos obtidos por meio dessa técnica. Por fim, as informações relevantes de cada estudo foram extraídas.

2.4 Mapeamento Sistemático: Fase 2 - Condução

O mapeamento sistemático foi conduzido por dois alunos de mestrado, com a colaboração de uma aluna de doutorado e de três especialistas. As etapas de obtenção dos estudos foram realizadas entre Setembro/2015 e Outubro/2015. As etapas de condução do mapeamento são detalhadas nas próximas seções.

2.4.1 Primeira Etapa

A identificação e a seleção dos estudos primários foram executadas de acordo com o protocolo estabelecido na Seção 2.3. As ferramentas Mendeley[1] e JabRef[2] foram adotadas para apoiar a condução do processo de seleção, organização e análise dos estudos primários. Para atender as especificidades das máquinas de busca, a *string* de busca foi adaptada como apresentado no Apêndice B.

No total, 2659 estudos primários foram retornados pelas máquinas de busca, sendo que 734 estudos duplicados foram excluídos e 1925 estudos foram selecionados para que fossem avaliados na primeira fase de seleção. Dos 1925 estudos, 88 estudos foram selecionados para a leitura completa na segunda fase.

Na Tabela 2.3, é apresentado um resumo da quantidade de estudos obtidos e incluídos separadamente para cada base. Na coluna "Obtidos" é apresentada a quantidade de estudos retornados por cada base. A coluna "Incluídos" mostra a quantidade de estudos incluídos após as atividades de seleção. Ressalta-se que essa quantidade considera os estudos repetidos entre as bases e, portanto, a soma dos valores dessa coluna é maior que o total de 88 estudos. Na coluna "% Incluídos", a taxa de estudos incluídos em relação a base é apresentada. Por fim, na coluna "% Impacto" é retratada a taxa de estudos selecionados na base em relação ao total de estudos selecionados.

[1] `www.mendeley.com`
[2] `www.jabref.org`

Tabela 2.3: Número de estudos incluídos por máquina de busca

Máquina de busca	Obtidos	Incluídos	% Incluídos	% Impacto
ACM Digital Library	87	10	11,49%	11,36%
IEEE Xplore	882	68	7,70%	77,27%
Science Direct	69	4	5,79%	4,54%
Scopus	1197	81	6,76%	92,04%
Web of Science	424	23	5,42%	26,13%

2.4.2 Segunda Etapa

Nessa etapa, foi realizada a leitura completa dos 88 estudos e os critérios de seleção foram novamente aplicados. Ao final da leitura, 18 estudos primários foram incluídos no mapeamento sistemático. Esses estudos são listados na Tabela 2.4, na qual são apresentados Ano, Título, Autores e um Código para referenciar cada estudo no decorrer deste relatório.

Tabela 2.4: Estudos incluídos utilizando as máquinas de busca

	Título	Autores	Código
1	Software Testing at the Architectural Level	Richardson e Wolf [41] (1996)	S01
2	Deriving Tests from Software Architectures	Ji e Offutt [22] (2001)	S05
3	Model-Checking Plus Testing: from Software Architecture Analysis to Code Testing	Bucchiarone et al. [8] (2004)	S06
4	Systematic Testing of Software Architectures in the C2 Style	Muccini et al. [32] (2004)	S04
5	Using Software Architecture for Code Testing	Muccini et al. [33] (2004)	S08
6	An Architecture-Centric Approach for Producing Quality Systems	Bertolino et al. [4] (2005)	S09
7	Software Architecture-based Regression Testing	Muccini et al. [31] (2006)	S11
8	Testability of Software in Service-oriented Architecture	Tsai et al. [46] (2006)	S12
9	Using Model Differencing for Architecture-Level Regression Testing	Muccini [30] (2007)	S14
10	A Method to Test Concurrent Systems Using Architectural Specification	Reza e Grant [39] (2007)	S15
11	Architecture Centric Approach to Enhance Software Testing Management	Li et al. [25] (2008)	S16
12	A Method to Generate Embedded Real-Time System Test Suites Based on Software Architecture Specifications	Ye et al. [51] (2008)	S17
13	Architecting Fault Tolerance with Exception Handling: Verification and Validation	Brito et al. [7] (2009)	S18
14	Test Methods of the AUTOSAR Application Software Components	Park et al. [37] (2009)	S20
15	An Automatic Generation Method of Executable Test Case Using Model-driven Architecture	Wang et al. [48] (2009)	S21
16	Design and Implementation of Automatic Generation of Test Cases Based on Model Driven Architecture	Yang et al. [50] (2010)	S23
17	Design of a Highly Adaptive Auto-test Software Platform Based on Common Data Interface	Hao et al. [13] (2010)	S24
18	Delta-Oriented Model-based Integration Testing of Large-Scale Systems	Lochau et al. [26] (2010)	S26

A técnica *backward snowballing* foi aplicada nos 18 estudos incluídos para identificar estudos relevantes que não foram retornados pelas máquinas de busca. Um total de oito estudos foram identificados, que são apresentados na Tabela 2.5. Além disso, o estudo de Leroux et al. [24] foi incluído no mapeamento sistemático por recomendação de especialistas. Ao final da segunda etapa, 27 estudos foram incluídos no mapeamento sistemático e serão utilizados para responder às questões de pesquisas.

Tabela 2.5: Estudos incluídos por *backward snowballing*

	Título	Autores	Código
1	Architecture-Based Regression Testing of Evolving Systems	Harrold [14] (1998)	S02
2	Challenges in Exploiting Architectural Models for Software Testing	Rosenblum [42] (1998)	S03
3	Testing Complex Architectural Conformance Relations	Young [52] (1998)	S04
4	Architectural Unit Testing	Scollo e Zecchini [44] (2005)	S10
5	Automated Generation of Test Cases Using Model-Driven Architecture	Javed et al. [21] (2007)	S13
6	Analyzing Relation between Software Architecture Testing Criteria on Test Sequences	Lun e Ding [27] (2009)	S19
7	Model Based Testing Using Software Architecture	Reza e Lande [40] (2010)	S22
8	A Model-Based Testing Framework for Automotive Embedded Systems	Marinescu et al. [29] (2014)	S27

2.5 Mapeamento Sistemático: Fase 3 - Apresentação dos Resultados

Nesta fase, foi utilizado o formulário apresentado no Apêndice C para extrair informações relevantes e responder às questões de pesquisa. Para apresentar os resultados, serão utilizadas tabelas compostas pelas colunas *categoria*, *quantidade de estudos (QE)* e *porcentagem de estudos (PO)*, que representam, respectivamente, a categoria em que estudos foram classificados, a quantidade de estudos e a porcentagem por categoria em relação aos 27 estudos.

2.5.1 RQ-1: Quais abordagens têm explorado arquitetura de software e teste de software em conjunto no processo de desenvolvimento de software?

Na análise dos 27 estudos incluídos, 18 estudos utilizam alguma abordagem para apoiar o teste de programas por meio da arquitetura de software. Os outros nove estudos não explicitam abordagens específicas. Três abordagens foram identificadas nos 18 estudos: teste baseado em modelos, testabilidade em SOA e teste baseado em MDA. Na primeira, a descrição da arquitetura de software foi utilizada para derivar modelos relevantes para a atividade de teste e, por meio desses modelos, definir casos de teste para aplicar na atividade de teste. Os estudos dessa abordagem têm direta relação com o teste baseado

em modelos. A testabilidade em SOA foi utilizada no estudo S12. Os autores utilizam o estilo arquitetural SOA para entender como um sistema pode ser testado e para derivar requisitos de teste [46]. Por fim, três estudos adotaram o teste baseado nas fases similares às que compõem a MDA, com o objetivo de estabelecer como casos de testes podem ser definidos.

Além das três abordagens, também foram identificadas as fases de teste e as técnicas de teste abordadas em cada estudo. A fase de teste de unidade foi abordada em cinco estudos, sendo que todos eles utilizam a técnica de teste funcional. O teste de integração foi utilizado em 11 estudos, com o uso da técnica de teste funcional. O teste de regressão foi observado em sete estudos: o estudo S20 adotou o teste estrutural e os demais a técnica funcional. Observa-se que somente os estudos S01 e S26 abordaram as três fases de teste, enquanto o estudo S11 as fases de teste de integração e regressão, como pode ser observado na Tabela 2.6. Um total de oito dos 27 estudos (S04, S06, S09, S15, S16, S17, S19, S27) não deixam claro qual fase de teste eles abordam.

Tabela 2.6: Abordagens que relacionam teste e arquitetura

Fase de Teste	Técnica de Teste	Baseado em Modelos	Testabilidade em SOA	Baseado em MDA
Unidade	Funcional	S01, S10, S18, S26	S12	-
	Estrutural	-	-	-
Integração	Funcional	S01, S05, S07, S08, S11, S24, S25, S26	-	S13, S21, S23
	Estrutural	-	-	-
Regressão	Funcional	S01, S02, S03, S11, S14, S26	-	-
	Estrutural	S20	-	-

Foram identificadas ainda informações arquiteturais que nortearam a atividade de teste nas abordagens identificadas, conforme Tabela 2.7. Essas informações retratam o modo como o teste de software foi relacionado à arquitetura de software. Dentre os trabalhos, 26% abordaram aspectos de testabilidade de sistemas de software. O teste de conformidade foi utilizado em 56% dos estudos e o teste de fluxo de dados da arquitetura foi considerado em 4% dos estudos. Os demais estudos, ou seja, 33%, não se encaixaram em nenhuma das características identificadas.

Os artefatos de teste produzidos nas abordagens foram distribuídos em sete categorias, sendo que a maioria dos estudos (78%) derivaram artefatos concretos, como casos de teste. Um estudo argumenta que a arquitetura pode ser utilizada para gerenciar a atividade de teste. Na Tabela 2.8, são apresentadas sete categorias de artefatos de teste, suas descrições, quantidades e porcentagens de estudos por categoria.

Tabela 2.7: Características gerais de teste

Categoria	Descrição	QE	PO
Testabilidade	A testabilidade de sistemas de software dada sua arquitetura.	7	26%
Estilo arquitetural	O estilo arquitetural é utilizado como fonte para derivar requisitos de teste.	2	7%
Teste de conformidade	No estudo, tem-se o objetivo de verificar a conformidade entre a arquitetura e o sistema implementado	15	56%
Teste utilizando fluxo de dados	O estudo utiliza o fluxo de dados entre elementos da arquitetura para derivar requisitos de teste.	1	4%
Não especificaram	O estudo não se enquadrou nas categorias identificadas.	9	33%

Tabela 2.8: Informações e artefatos de teste gerados nos estudos

Artefato	Descrição	QE	PO
Casos de teste	Casos de teste são selecionados ou derivados a partir da arquitetura do sistema em teste.	21	78%
Critério de teste	São definidos critérios de cobertura e funcionalidades para partes específicas de uma arquitetura.	8	30%
Critério de adequação	Critérios são definidos para verificar se um conjunto de casos de teste é adequado para um sistema.	2	7%
Sequências de teste	Modelos da arquitetura são utilizados para derivar sequências de teste que poderão ser utilizadas em modelos.	5	19%
Gerenciar atividade de teste	A arquitetura de software é utilizada para contribuir para o gerenciamento da atividade de teste.	1	4%
Oráculo de teste	A arquitetura de software é utilizada para derivar oráculos para a atividade de teste.	1	4%
Plano de teste	A arquitetura é utilizada como apoio ao desenvolvimento de um plano de teste.	3	11%

Para sumarizar as informações das classificações apresentadas nesta seção, foi criada a Tabela 2.9, adaptada de Muccini [30], que contém uma comparação entre os estudos e as categorias identificadas.

2.5.2 RQ-2: Quais informações arquiteturais têm sido consideradas para apoiar a atividade de teste?

Na segunda questão de pesquisa, buscou-se identificar como as informações arquiteturais foram obtidas e descritas no escopo das abordagens identificadas na RQ-1. Nos 27 estudos incluídos, foram identificadas quatro fontes de informações arquiteturais para apoiar a atividade de teste, como ilustrado na Tabela 2.10.

A análise estática foi utilizada por 70% dos estudos e refere-se à análise de elementos em tempo de projeto e sua organização [43], por exemplo, a utilização de modelos arquiteturais

Tabela 2.9: Tabela comparativa entre estudos com relação ao teste de software

	Tópicos de teste	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27
Atividade de teste	Casos de teste		✓			✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
	Critério de teste	✓	✓			✓			✓				✓							✓			✓					✓
	Critério de adequação			✓					✓																			
	Sequências de teste					✓			✓							✓			✓	✓								
	Gerenciar atividade de teste																✓											
	Gerar oráculo de teste				✓																							
	Gerar plano de teste	✓							✓														✓					
Fases	Unidade	✓									✓		✓						✓								✓	
	Integração	✓				✓		✓	✓			✓				✓			✓				✓		✓	✓	✓	
	Regressão	✓	✓	✓								✓			✓												✓	
Técnica	Funcional	✓					✓	✓	✓	✓	✓	✓			✓	✓			✓	✓	✓		✓			✓	✓	✓
	Estrutural																				✓							
Objetivo	Teste baseado em arquitetura	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Validação de arquitetura	✓					✓			✓									✓									
Outros	Testabilidade	✓	✓								✓		✓				✓				✓		✓					
	Estilo arquitetural							✓					✓															
	Teste de conformidade	✓			✓	✓	✓	✓	✓	✓	✓	✓			✓				✓		✓		✓			✓	✓	
	Fluxo de dados					✓																						

Tabela 2.10: Informações da arquitetura de software utilizadas

Fonte de Informação	Estudos	QE	PO
Análise Estática	S01, S02, S03, S04, S06, S07, S09, S10, S11, S12, S14, S15, S17, S18, S19, S21, S23, S24, S26, S27	20	70%
Análise Dinâmica	S01, S05, S06, S07, S08, S09, S10, S16, S18, S19, S20, S23, S26, S27	14	52%
Transformações de Modelos MDA	S13, S22, S23	3	11%
Gerenciamento de Configuração	S02, S11, S14	3	11%

como base para planejar o teste de unidade e de integração de grandes sistemas [26]. A estrutura e a descrição desses elementos são utilizadas como fontes de informação para a atividade de teste. Na análise dinâmica, as relações dos elementos e suas interações são consideradas como fontes de informação para a atividade de teste. Por exemplo, a troca de mensagens entre elementos da arquitetura ou, até mesmo, a especificação das interfaces. O método de transformação de modelos de MDA foi adotado por 11% dos estudos para definir casos de teste abstratos e, posteriormente, refinados para casos de teste concretos por meio de transformações entre o PIM e o PSM. Além disso, em 11% dos estudos, o gerenciamento de configuração da documentação arquitetural atua como apoio para selecionar casos de teste para o teste de regressão. Os estudos S01, S02, S06, S07, S09, S10, S11, S12, S14, S18, S19, S23, S26, e S27 utilizaram mais de uma fonte de

informação arquitetural no apoio a atividade de teste. Por exemplo, o estudo S09 utiliza as fontes de informações estática e dinâmica. Além disso, para representar as arquiteturas, diferentes linguagens foram utilizadas, conforme mostra a Tabela 2.11.

Tabela 2.11: Linguagens utilizadas para representar a arquitetura

Linguagem	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27
CHAM	✓	✓	✓																✓								
ADL Wright					✓																						
LTS							✓	✓																			
π-ADL																									✓		
HPrTNs															✓							✓					
UML									✓	✓		✓		✓				✓			✓		✓				
B-METHOD																		✓									
EAST-ADL																											✓
MEF				✓		✓			✓		✓															✓	
IOLTS										✓															✓		
Basic LOTOS										✓																	
EMF												✓															
DRTSADL																	✓										
TIOA																	✓										
ACME ADL																						✓					
Model Delta																										✓	
ADL																										✓	

Legenda: CHAM - *Chemical Abstract Machine*; LTS - *Labeled Transition System*; HPrTNs - *Hierarchical Predicate Transition Nets*; UML - *Unified Modeling Language*; DRTSADL - *Architecture description language of distributed/embedded real-time system*; IOLTS - *Input/Output Labelled Transition Systems*; EMF - *Eclipse Modeling Framework*; TIOA - *Timed Input/Output Automaton*; ADL - *Architecture Description Language*; MEF - Máquina de Estados Finita.

A linguagem mais utilizada foi a UML, adotada por nove estudos. Os estudos S9, S10, S12, S17, S18, S22, S25 e S26 adotaram mais de uma linguagem para representar a arquitetura. Um exemplo é o estudo S09 que considera as fontes de informações estática e dinâmica e, em função disso, utiliza a UML e MEF para representar a arquitetura.

2.5.3 RQ-3: Existem ferramentas que oferecem suporte às abordagens identificadas?

Dada a importância da automatização no desenvolvimento de sistemas de software com qualidade, essa questão de pesquisa tem o objetivo de identificar ferramentas relacionadas às abordagens da RQ-1. Dos estudos selecionados para extração de dados, apenas cinco utilizaram ferramentas para auxiliar a automatização de suas abordagens.

A abordagem proposta em S08 consiste em explorar a dinâmica de uma arquitetura de software para identificar interações entre os componentes do sistema e criar classes de testes para os comportamentos arquiteturais mais relevantes [33]. Seus autores propõem modelar o comportamento arquitetural na linguagem LTS[3] (*Labeled Transition Systems*)

[3] `http://www.mcrl2.org/dev/user_manual/articles/lts.html`

e, a partir disso, derivar um conjunto adequado de abstrações do modelo LTS denominadas ALTSs (*Abstract Labeled Transition Systems*). No estudo de caso apresentado, os autores utilizam uma ferramenta denominada LTSA[4] (*Labeled Transition System Analyzer*) para derivar estados em linguagem LTS recebendo como entrada uma especificação em FSP[5] (*Finite State Process*).

A abordagem proposta por S10 consiste em modelar unidades arquiteturais em diagramas de estado traduzidas para linguagem *Basic* LOTOS (*Language of Temporal Ordering Specification*) [44]. A partir de uma especificação descrita na linguagem Basic LOTOS, os autores utilizam a ferramenta TGV (*Test Generation using Verification techniques*) para gerar casos de teste e verificar a conformidade de uma unidade de arquitetura com as funcionalidades necessárias.

Na abordagem proposta por S11, os autores exploram como o teste de regressão pode ser aplicado em nível arquitetural. Os autores verificam se uma implementação ligeiramente modificada está em conformidade com a arquitetura inicial [31]. No estudo de caso, um conjunto de ferramentas foi adotado em diferentes etapas. Inicialmente, os autores especificaram a arquitetura no estilo arquitetural C2 e, a partir da especificação, conduziram dois experimentos [31]: um utilizando a ferramenta FC2 *Tools* para determinar um ATS (*Abstract Transition System*) e outro a ferramenta LTSA para derivar estados na linguagem LTS.

Na abordagem definida em S14, é analisado como uma especificação arquitetural baseada em modelos pode ser utilizada para implementar teste de regressão em nível arquitetural [30]. Para selecionar um conjunto de casos de teste a partir de uma especificação arquitetural, os autores utilizam um *plugin* para a ferramenta CHARMY[6] (*CHecking ARchitectural Model consistencY*). Esse *plugin*, a partir de um diagrama de estado de componente, implementa a cobertura de vários critérios (por exemplo, todas arestas, todos componentes, todos estados e McCabe).

Por fim, a abordagem proposta em S19 consiste em derivar uma sequência de teste a partir de uma arquitetura descrita em CHAM. Nela, os autores aplicam a ferramenta RDG (*Rule Dependence Graph*) para criar um grafo do sistema em teste. A partir desse grafo, um conjunto de regras de cobertura é gerado para determinar uma sequência de teste para ser aplicada no programa [27].

[4] `http://www.doc.ic.ac.uk/ltsa/`
[5] `http://www.doc.ic.ac.uk/~jnm/LTSdocumention/FSP-notation.html`
[6] `http://www.di.univaq.it/charmy/`

2.6 Considerações Finais

Neste capítulo, foram apresentados o protocolo, a condução e os resultados obtidos no mapeamento sistemático. A arquitetura de software e o teste software têm sido relacionados por meio do teste baseado em arquitetura. A técnica mais utilizada, sendo considerada em 18 estudos entre os 27, é a funcional. A técnica estrutural foi considerada em apenas um estudo e não houve referência à técnica baseada em defeitos. Considerando as fases de teste, 11 estudos adotaram o teste de integração, seguido pelo teste de unidade com cinco estudos. A arquitetura de software foi considerada para aprimorar a testabilidade de sistemas. Os artefatos de teste gerados a partir das abordagens foram, majoritariamente, casos de teste (78%), seguido por critérios de teste (30%), sendo que oráculos foram considerado apenas por um estudo.

Apesar da arquitetura ser utilizada para apoiar a atividade de teste, em apenas quatro estudos foi considerada a atividade de avaliação da arquitetura de software. Contudo, em 56% dos estudos considerou-se verificar a conformidade entre a arquitetura de software e o sistema implementado. Em relação às fontes de informações arquiteturais, as análises estática e dinâmica foram as mais utilizadas nos estudos, seguidas pelo gerenciamento de configuração de versões da arquitetura e transformação de modelos MDA. Para representar a arquitetura, diferentes linguagens foram utilizadas, sendo que a linguagem mais utilizada foi a UML.

Apesar da importância da automatização no processo de desenvolvimento de software, apenas cinco estudos utilizaram ferramentas, o que indica a necessidade de maiores esforços nesse sentido.

Capítulo 3

Visão Geral da Área de Pesquisa

3.1 Considerações Iniciais

Neste capítulo é apresentada uma visão geral do tópico de pesquisa abordado neste mapeamento sistemático. Dessa forma, este capítulo está organizado da seguinte forma. Na Seção 3.2, é descrita a relação entre a quantidade de estudos e os anos em que elas foram publicadas. Na Seção 3.3, são apresentados os tipos de avaliações conduzidas nos estudos selecionados. Na Seção 3.4, são listados os países dos autores dos estudos selecionados. Na Seção 3.5 são apresentadas as fases e as técnicas de teste que foram relatadas nos estudos. Na Seção 3.6 são mostradas as relações entre os autores por meio das publicações em comum. Na Seção 3.7, apresentam-se os veículos em que os estudos foram publicados. Na Seção 3.8, um *wordcloud* é apresentado. Na Seção 3.9, apresentam-se as ameaças à validade para esse mapeamento. Por fim, na Seção 3.10, são relatadas as considerações finais deste capítulo.

3.2 Quantidade de Estudos Publicados por Ano

A análise de publicações por ano permite identificar tendências na área ou apontar uma redução na quantidade de publicações. Na Figura 3.1 é apresentada a quantidade de estudos publicados entre os anos de 1996 e 2014. Pode-se perceber que as publicações sobre arquitetura de software e teste de software tiveram início em 1996. Além disso, o ano de 2009 obteve o maior número de estudos publicados e os anos de 1996, 1998, 2011 e 2013 a menor quantidade de estudos.

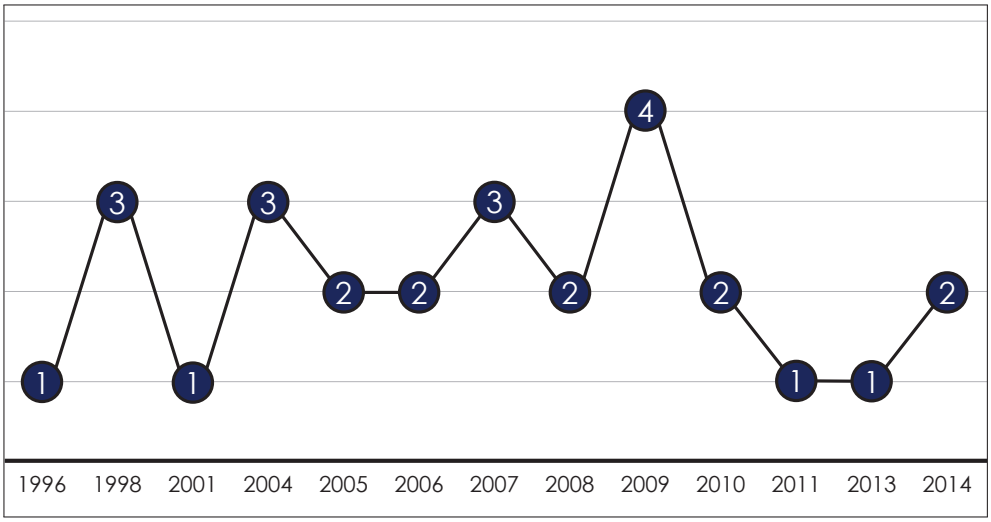

Figura 3.1: Quantidade de estudos por ano

3.3 Tipos de Avaliações

A análise das avaliações conduzidas nos estudos fornece uma visão de como os estudos foram realizados e a maturidade do tópico de pesquisa. Nesse mapeamento, foram identificados dois tipos de avaliação empírica: estudo de caso e experimento. Um total de 63% dos estudos passaram por um dos tipos de avaliação. Entre esses estudos, 56% utilizaram o estudo de caso e 7% experimento, como pode ser visto na Tabela 3.1. Um total de 37% dos estudos não realizaram avaliações.

Tabela 3.1: Tipo de avaliação das abordagens propostas

Tipo de avaliação	Estudos	QE	PO
Estudo de Caso	S06, S07, S10, S11, S12, S13, S14, S17, S18, S20, S22, S24, S25, S26, S27	15	56%
Experimento	S05, S16	2	7%
Sem Avaliação	S01, S02, S03, S04, S08, S09, S15, S16, S21, S23	10	37%
	Total	27	100%

Quanto ao ambiente em que os estudos foram conduzidos, tem-se que 93% deles foram conduzidos no ambiente acadêmico e 7% na indústria, como ilustrado na Tabela 3.2.

Tabela 3.2: Contexto em que os estudos foram conduzidos

Contexto	Estudos	QE	PO
Academia	S01, S02, S03, S04, S07, S08, S09, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27	25	93%
Indústria	S05, S06	2	7%
	Total	27	100%

3.4 Autores por País

A análise dos autores permite identificar os países em que se encontram os principais pesquisadores da área. Os 27 estudos possuem um total de 66 pesquisadores envolvidos, de 32 universidades em 11 países. No Apêndice A, são listados todos os autores, suas instituições e países. A distribuição de estudos por país é ilustrada na Figura 3.2.

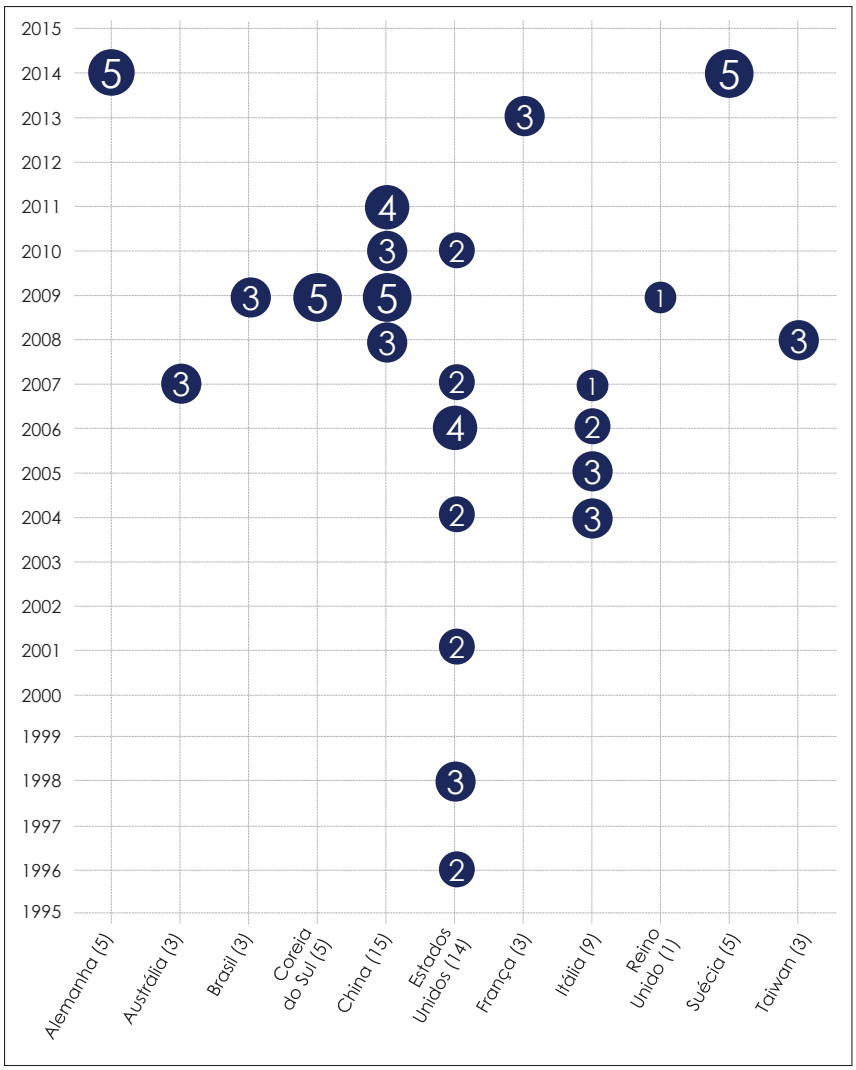

Figura 3.2: Quantidade de autores distribuídos por país e por ano

O país com a maior quantidade de pesquisadores é a China, que possui 15 pesquisadores, seguida pelos Estados Unidos com 14 e depois pela Itália com nove.

3.5 Técnicas e Fases de Teste

Na Figura 3.3 é mostrada a relação entre a quantidade de estudos obtidos, fases de teste (à direita) e técnicas de teste (à esquerda).

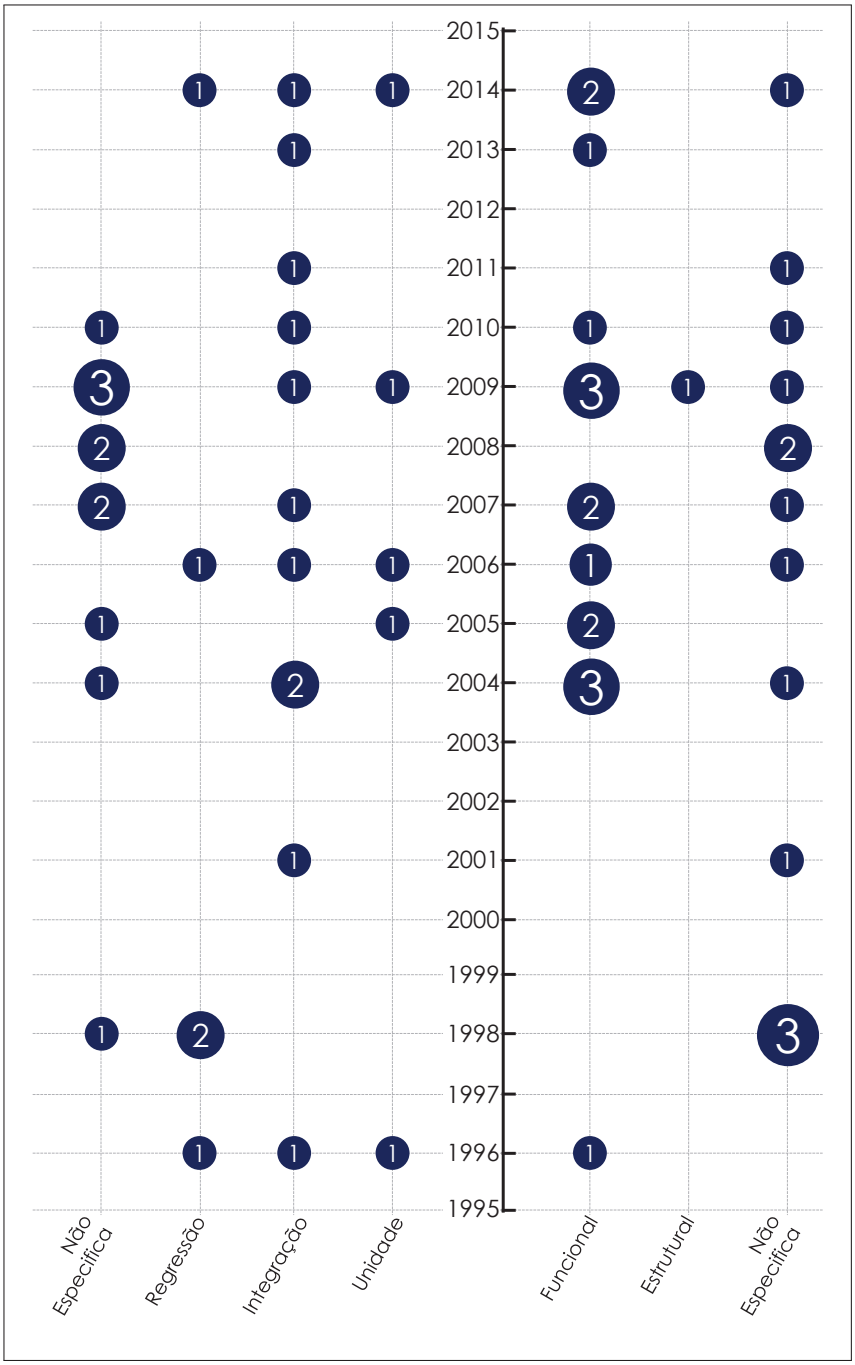

Figura 3.3: Fases de teste e técnicas de teste

A partir de 2004 houve um aumento na quantidade de publicações e os estudos voltam a cobrir mais que uma fase de teste. Os estudos não se referiram a fase de teste de sistema e nem a técnica de teste baseado em defeitos, a técnica de teste estrutural foi referenciada em apenas um estudo. Reitera-se que a fase de teste mais utilizada foi a de integração e a técnica de teste foi a funcional.

3.6 Relacionamento entre Autores

A análise de como os autores têm interagido entre eles permite compreender quais são os autores que mais pesquisam na área, além disso, é complementar a análise de autores por país. Na Figura 3.4 é apresentado o grafo que relaciona autores e estudos publicados. Nesse grafo, foram considerados como grupos de pesquisa os autores que têm alguma publicação em comum. Na figura, os autores são representados pelas células brancas e os estudos as células pretas.

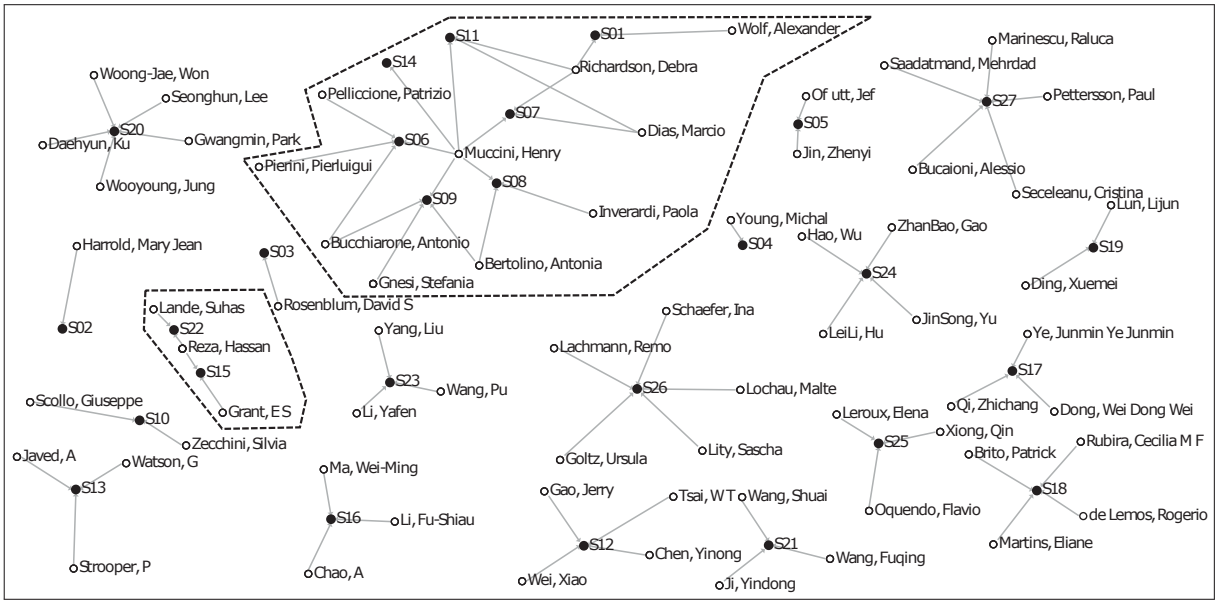

Figura 3.4: Grafo da relação entre autores e estudos publicados.

É possível notar que apenas dois grupos possuem mais que uma publicação, esses estão destacados na figura. Foram identificados dois autores com mais de duas publicações. Um deles, da *Universita' di L'Aquila* da Itália, colaborou em seis estudos, a saber S06, S07, S08, S09, S11 e S14. Outro autor, da *University of California Irvine* nos Estados Unidos, colaborou com os estudos S01, S07 e S11.

3.7 Relacionamento entre Veículos de Publicação e Avaliações

Nesta seção, são apresentados os tipos de veículos de publicação utilizados pelos 27 estudos em relação aos tipos de avaliações descritos na Seção 3.3. Uma lista de todos os veículos utilizados pelos estudos é apresentada no Apêndice D. No mapa da Figura 3.5, os tipos de veículos de publicação são apresentados à esquerda e os tipos de avaliações à direita. A maioria dos estudos foram publicados em conferências e avaliados por meio de estudos de caso.

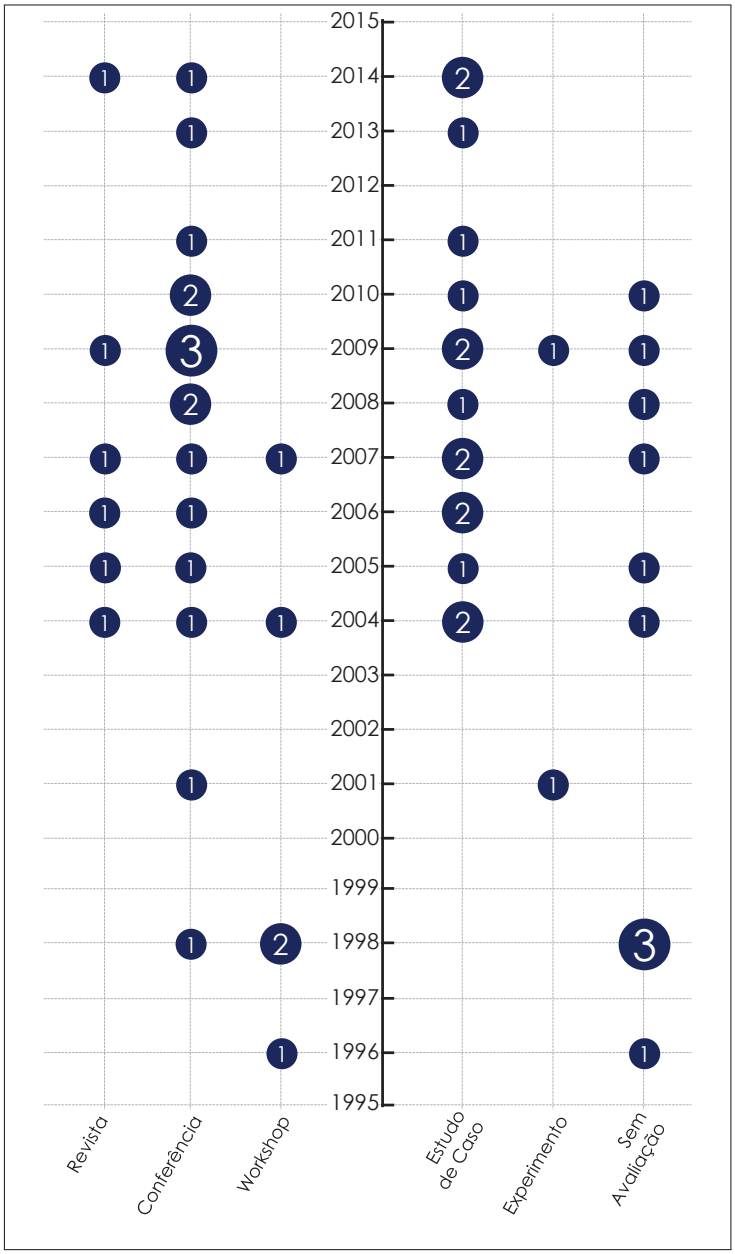

Figura 3.5: Tipos de veículo de publicação e tipo avaliações conduzidas

O maior número de publicações está entre os anos de 2008 e 2010, em que foram publicados sete estudos, sendo que, desses, três não foram avaliados. O número de publicações na área está diminuindo nos últimos anos e a preocupação com a avaliação tem aumentado, sendo que nos anos de 2013 e 2014 todos os estudos foram avaliados por meio de estudos de caso.

3.8 *Wordcloud*

Para identificar as palavras mais utilizadas nos 27 estudos, foi criado a *wordcloud* com base nos resumos dos estudos e apresentado na Figura 3.6. Por meio dessa *wordcloud*,

obtêm-se uma visão geral dos termos mais utilizados no resumos. Ela também auxiliou no processo de avaliação da *string* de busca. Pode-se perceber que os termos relacionados a teste de software e arquitetura de software tiveram bastante destaque; contudo, outros também sobressaíram, tais como *"architecture-based"*, *"model"*, e *"formal"*.

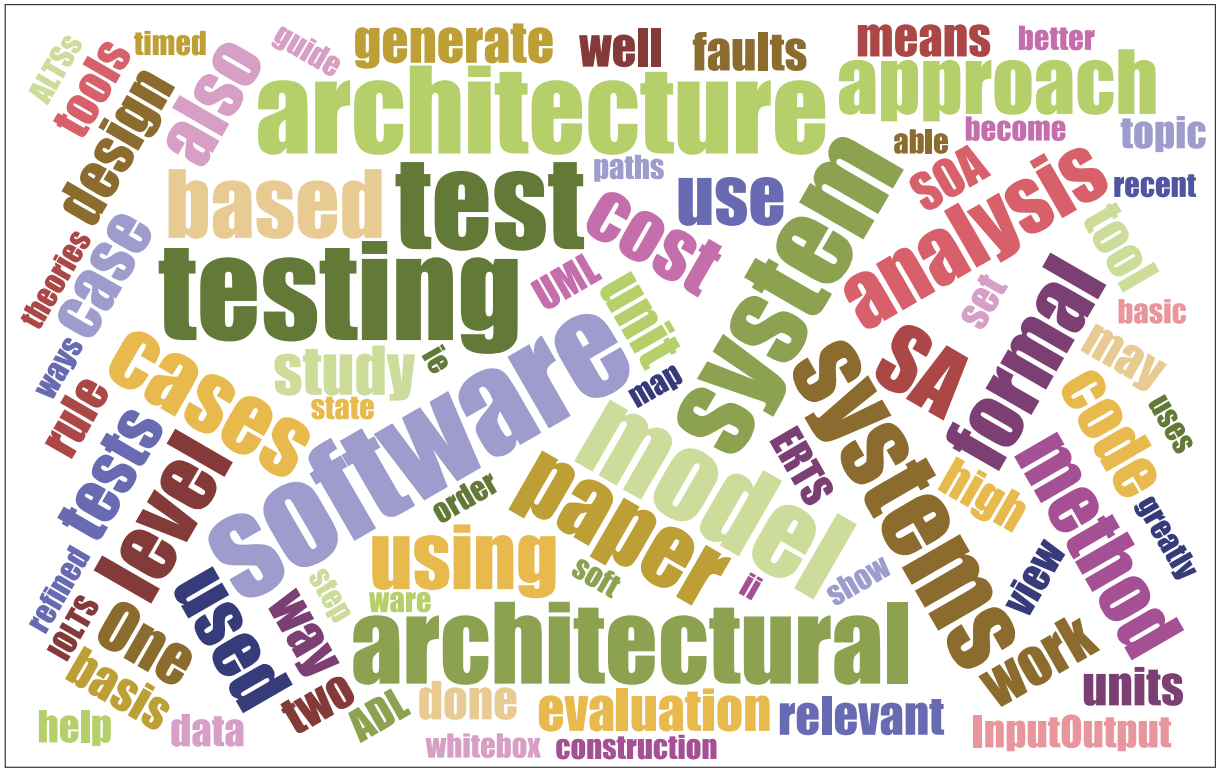

Figura 3.6: *Wordcloud* das palavras contidas nos resumos dos estudos

3.9 Ameaças à Validade

Na condução desse mapeamento sistemático, as seguintes ameaças à validade foram identificadas e discutidas e, eventualmente mitigadas.

- Perda de estudos importantes: durante a pesquisa, é possível que estudos importantes possam não ter sido sido incluídos. Há problemas recorrentes em revisões sistemáticas e mapeamentos sistemáticos, tais como [28]: (i) identificar bons estudos; (ii) extração correta de dados; e (iii) classificação adequada de cada estudo. Para identificar estudos relevantes, ou seja, aqueles com maior potencial para responder as questões de pesquisa, os autores selecionaram as bases de busca baseando-se na sugestão de especialistas e nas recomendações de Kitchenham e Charters [23]. No entanto, não foram consideradas estudos da literatura cinzenta e nem outras fontes de buscas existentes;

- *String* de busca: a *string* de busca foi definida com o auxílio de especialistas. No entanto, os termos teste de software e arquitetura de software são utilizados e referenciados em diferentes contextos mesmo que não tenham sido efetivamente abordados nos estudos; Apenas 1% dos estudos selecionados foram considerados na extração de dados. Nesse caso, a estratégia adotada foi deixar a *string* genérica e excluir os estudos não relevantes manualmente;

- Apresentação dos resultados: em alguns estudos poucos detalhes sobre as abordagens propostas foram apresentados. Há falta de clareza sobre como algumas contribuições foram feitas e como os estudos foram realizados, dificultando, assim, a extração dos dados para responder as questões de pesquisa; e

- Experiência dos autores: dois dos pesquisadores que conduziram esse mapeamento são pós-graduandos em nível de mestrado nas áreas de teste de software e arquitetura de software. Sendo assim, as principais decisões eram tomadas com o apoio de especialistas em ambas as áreas (teste de software e arquitetura de software) e em mapeamentos sistemáticos.

3.10 Considerações Finais

Neste capítulo, foi apresentada uma visão geral dos resultados encontrados no mapeamento. Desde o primeiro estudo publicado em 1996, passaram-se 20 anos, sendo que, a maioria dos estudos concentram-se na segunda década. A quantidade de estudos publicados por ano ainda é pequena e as informações descritas nesses estudos não são detalhadas suficientemente. Apenas seis estudos foram publicações em revistas, sendo a maioria (16) em conferências e cinco em *workshops*.

Foram destacados os tipos de avaliações experimentais conduzidos. Somente 17 dos 27 estudos (63%) tiveram alguma avaliação, predominantemente por meio estudos de caso. Praticamente os estudos foram realizados no ambiente acadêmico, com exceção de dois deles realizados na indústria: um estudo de caso e um experimento. Fica evidente a necessidade de condução de avaliações experimentais para maior consolidação e maturidade das iniciativas.

Por meio de um grafo de autoria, buscou-se apresentar um panorama a respeito de grupos de pesquisa. Foi identificado que apenas dois grupos de pesquisa publicaram mais que um estudo. As publicações são de autores oriundos de 15 países, sendo que a maioria deles são da China e dos Estados Unidos.

Capítulo 4

Conclusões e Perspectivas

Neste trabalho, conduziu-se um mapeamento sistemático para categorizar o estado da arte considerando o relacionamento entre arquitetura de software e teste de software no processo de desenvolvimento de software. Foram identificados poucos trabalhos que relacionam arquitetura e teste de software, dada a relevância dessas áreas para o desenvolvimento de software. Arquitetura de software é fundamental para alcançar a qualidade desejada e facilitar na evolução de grandes sistemas. Teste de software também é uma atividade fundamental nesse contexto. Processos bem estabelecidos e com alta qualidade precisam abordar sistematicamente essas áreas, para que os produtos sejam desenvolvidos com a qualidade adequada para o fim que são destinados.

Na condução desse mapeamento foram adotados o processo proposto por Kitchenham e Charters [23], a técnica *backward snowballing* [49] e as indicações de Petersen et al. [38], considerando estudos publicados até novembro de 2015. Ao todo, foram identificados e selecionados 27 estudos relevantes para o tópico de pesquisa. Notou-se que todos os estudos identificados tinham o objetivo de fornecer bases para o teste de software a partir das informações da arquitetura, caracterizado-se como teste baseado em arquitetura. As principais fontes de informações arquiteturais para o teste de software foram as análises estática e dinâmica de modelos arquiteturais. Para realizar as descrições arquiteturais, diversas linguagens foram utilizadas, sendo que a UML foi a linguagem mais considerada.

Embora em 63% dos estudos foram conduzidas avaliações, há uma falta de evidências sobre a eficiência e a eficácia dos métodos/técnicas propostos. Dos 27 estudos, 21 foram publicados em conferências ou *workshops* e somente 6 foram publicados em revistas. Adicionalmente, apenas dois estudos foram aplicados na indústria, os demais foram conduzidos na academia. Desse cenário, fica evidente a necessidade de condução de avaliações experimentais para maior consolidação e maturidade das iniciativas.

Conforme destacado anteriormente, a arquitetura de software e o teste software têm sido investigados por meio do teste baseado em arquitetura. Entretanto, apenas algumas

fases e técnicas têm sido investigadas. A técnica funcional é a mais utilizada, seguida da técnica estrutural, considerada em apenas um estudo. Não houve referência à técnica baseada em defeitos. Considerando as fases de teste, 11 estudos adotaram o teste de integração, seguido pelo teste de unidade com cinco estudos. A arquitetura de software foi considerada para aprimorar a testabilidade de sistemas em sete estudos. Os artefatos de teste gerados a partir das abordagens foram, majoritariamente, casos de teste (78%), seguido por critérios de teste (30%), sendo que oráculos foram considerado apenas por um estudo.

Apesar da arquitetura ser utilizada para apoiar a atividade de teste, em apenas quatro estudos foi considerada a atividade de avaliação da arquitetura de software. Contudo, em 56% dos estudos, considerou-se verificar a conformidade entre a arquitetura de software e o sistema implementado. Em relação às fontes de informações arquiteturais, as análises estática e dinâmica foram as mais utilizadas nos estudos, seguidas pelo gerenciamento de configuração de versões da arquitetura e transformação de modelos MDA.

Apesar da importância da automatização no processo de desenvolvimento de software, apenas cinco estudos utilizaram ferramentas, o que indica a necessidade de maiores esforços nesse sentido. As pesquisas nessa área são recentes, desde o primeiro estudo publicado decorreram-se 20 anos, sendo que a maior parte dos estudos está concentrada nos últimos 10 anos. A quantidade de estudos publicados por ano ainda é pequena e as informações descritas nesses estudos não são detalhadas suficientemente. As publicações são de autores oriundos de 15 países, sendo que a maioria deles são da China e dos Estados Unidos. A partir de um grafo que relacionava autores e publicações, identificou-se que apenas dois grupos de pesquisa publicaram mais que um estudo.

Conclui-se que o uso complementar de informações de arquitetura de software e de teste de software encontra-se ainda em um estágio inicial, com oportunidades e desafios de pesquisa para a proposição de abordagens que visem a contribuir para a produção de sistemas de software de alta qualidade.

Nesse contexto, no escopo das atividades do grupo de Engenharia de Software do ICMC-USP, tem-se desenvolvido dois artefatos de software: o ProSA-RA (*Process Based on Software Architecture - Reference Architecture*) [36] e o RAModel (*Reference Achitecture Model*) [35]. Motivados pelos resultados obtidos nesse mapeamento, esses artefatos serão evoluídos de forma a possibilitar a incorporação de informações de teste de software durante o processo de estabelecimento de arquiteturas de software. As informações encapsuladas seriam úteis, com diretrizes apropriadas, para derivar artefatos de teste e orientar a atividade de teste desde os estágios iniciais do processo de desenvolvimento, na perspectiva da produção e evolução de produtos de software de alta qualidade.

Referências

[1] Wasif Afzal, Snehal Alone, Kerstin Glocksien e Richard Torkar. "Software test process improvement approaches: A systematic literature review and an industrial case study". Em: *Journal of Systems and Software* 111 (2016), pp. 1 –33.

[2] Len Bass, Paul Clements e Rick Kazman. *Software architecture in pratice*. 3ª ed. Addison–Wesley Professional, 2012, p. 640.

[3] Antonia Bertolino e Paola Inverardi. "Architecture–based software testing". Em: *Joint proceedings of the 2nd international software architecture workshop (ISAW 1996)*. São Francisco, Estados Unidos, 1996, pp. 62–64.

[4] Antonia Bertolino, Antonio Bucchiarone, Stefania Gnesi e Henry Muccini. "An architecture-centric approach for producing quality systems". Em: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 3712 (2005), pp. 21–37.

[5] Antonia Bertolino, Paola Inverardi e Henry Muccini. "Software architecture-based analysis and testing: a look into achievements and future challenges". Em: *Computing* 95.8 (2013), pp. 633–648.

[6] Marco Brambilla, Jordi Cabot e Manuel Wimmer. *Model-Driven Software Engineering in Practice*. 1ª ed. Morgan & Claypool Publishers, 2012, p. 182.

[7] Patrick Brito, Rogério De Lemos, Cecília Rubira e Eliane Martins. "Architecting fault tolerance with exception handling: Verification and validation". Em: *Journal of Computer Science and Technology* 24.2 (2009), pp. 212–237.

[8] Antonio Bucchiarone, Henry Muccini, P Pelliccione e P Pierini. "Model-checking plus testing: from software architecture analysis to code testing". Em: *Artificial Intelligence and Lecture Notes in Bioinformatics*. Toledo, Espanha, 2004, pp. 351–365.

[9] Paul Clements, David Garlan, Len Bass, Judith Stafford, Robert Nord, James Ivers e Reed Little. *Documenting software architectures: views and beyond*. 2ª ed. Pearson Education, 2010, p. 537.

[10] Márcio Eduardo Delamaro, José Carlos Maldonado e Mário Jino. *Introdução ao teste de software*. 2ª ed. Campus, 2016, p. 408.

[11] George Fairbanks. *Just enough software architecture: a risk–driven approach*. 1ª ed. Marshall & Brainerd, 2010, p. 376.

[12] David Garlan. "Software Architecture: A Roadmap". Em: *Conference on The Future of Software Engineering*. Nova York, Estados Unidos, 2000, pp. 91–101.

[13] Wu Hao, Gao ZhanBao, Yu Jinsong e Hu LeiLi. "Design of a highly adaptive auto-test software platform based on common data interface". Em: *Proceedings of the 6th IEEE Conference on Industrial Electronics and Applications (ICIEA 2011)*. Beijing, China, 2011, pp. 254–258.

[14] Mary Jean Harrold. "Architecture-Based Regression Testing of Evolving Systems". Em: *Proceedings of the International Workshop on the Role of Software Architecture in Testing and Analysis (Rosatea 1998)*. Sicília, Italia, 1998, pp. 73–77.

[15] Mary Jean Harrold. "Testing: A Roadmap". Em: *Conference on The Future of Software Engineering (ICSE 2000)*. Limerick, Ireland: ACM, 2000, pp. 61–72. ISBN: 1-58113-253-0.

[16] Uwe van Heesch, Paris Avgeriou e Rich Hilliard. "A documentation framework for architecture decisions". Em: *Journal of Systems and Software* 85.4 (2012), pp. 795–820.

[17] Christiane Hofmeister, Philippe Kruchten, Robert L Nord, Henk Obbink, Alexander Ran e Pierre America. "A general model of software architecture design derived from five industrial approaches". Em: *Journal of Systems and Software* 80.1 (2007), pp. 106–126.

[18] ISO/IEC/IEEE-29119-1. *Software and systems engineering – Software testing – Part 1: Concepts and definitions. Standard 29119-1:2013(E), International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE)*. 2013.

[19] ISO/IEC/IEEE:24765. *Systems and software engineering – Vocabulary*. 2010.

[20] ISO/IEC/IEEE:42010. *Systems and software engineering–Architecture description*. 2011.

[21] A Z Javed, Paul Anthony Strooper e Geoffrey N Watson. “Automated Generation of Test Cases Using Model-Driven Architecture”. Em: *International Workshop on Automation of Software Test (AST 2007)*. Washington, Estados Unidos, 2007, p. 3.

[22] Yindong Ji e Jeff Offutt. “Deriving tests from software architectures”. Em: *International Symposium on Software Reliability Engineering (ISSRE 2001)*. Hong kong, China, 2001, pp. 308–313.

[23] Barbara Kitchenham e Stuart Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Rel. téc. EBSE 2007-001. Joint Report Keele University e Durham University, 2007.

[24] Elena Leroux, Flavio Oquendo e Qin Xiong. “Architecture-Based Conformance Testing”. Em: *International Conference on Software Engineering Advances (ICSEA 2013)*. Veneza, Italia, 2013, pp. 55–64.

[25] Fu-Shiau Li, Wei-Ming Ma e Architect Chao. “Architecture centric approach to enhance software testing management”. Em: *International Conference on Intelligent Systems Design and Applications (ISDA 2008)*. Taiwan, China, 2008, pp. 654–659.

[26] Malte Lochau, Sascha Lity, Remo Lachmann, Ina Schaefer e Ursula Goltz. “Delta-oriented model-based integration testing of large-scale systems”. Em: *Journal of Systems and Software* 91 (2014), pp. 63–84.

[27] Lijun Lun e Xuemei Ding. “Analyzing Relation between Software Architecture Testing Criteria on Test Sequences”. Em: *IEEE International Conference on Secure Software Integration and Reliability Improvement (SERE-C 2009)*. Shanghai, China, 2009, pp. 181–186.

[28] Lech Madeyski, Wojciech Orzeszyna, Richard Torkar e Mariusz Jozala. “Overcoming the Equivalent Mutant Problem: A Systematic Literature Review and a Comparative Experiment of Second Order Mutation”. Em: *IEEE Transactions on Software Engineering* 40.1 (2014), pp. 23–42.

[29] Raluca Marinescu, Mehrdad Saadatmand, Alessio Bucaioni, Cristina Seceleanu e Paul Pettersson. “A Model-Based Testing Framework for Automotive Embedded Systems”. Em: *Proceedings of the 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2014)*. Verona, Italia, 2014, pp. 38–47.

[30] Henry Muccini. “Using Model Differencing for Architecture-level Regression Testing”. Em: *EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA’07)*. Luebeck, Alemanha, 2007, pp. 59–66.

[31] Henry Muccini, Marcio Dias e Debra Richardson. “Software architecture-based regression testing”. Em: *Journal of Systems and Software* 10.79 (2006), pp. 1379–1396.

[32] Henry Muccini, Marcio Dias e Debra Richardson. “Systematic testing of software architectures in the C2 style”. Em: *Fundamental Approaches to Software Engineering*. Barcelona, Espanha, 2004, pp. 295–309.

[33] Henry Muccini, Paola Inverardi e Antonia Bertolino. “Using software architecture for code testing”. Em: *IEEE Transactions on Software Engineering* 30.3 (2004), pp. 160–171.

[34] Glenford J Myers, Corey Sandler e Tom Badgett. *The art of software testing*. 3ª ed. John Wiley & Sons, 2011, p. 240.

[35] Elisa Yumi Nakagawa, Flávio Oquendo e Martin Becker. “RAModel: A Reference Model for Reference Architectures”. Em: *Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*. Helsinki, Finlândia: IEEE, 2012, pp. 297–301.

[36] Elisa Yumi Nakagawa, Flavio Oquendo e José Carlos Maldonado. “Reference architecture”. Em: *Software Architecture 1*. Ed. por Mourad Chabane Oussalah. 1ª ed. John Wiley & Sons, Ltd, 2014. Cap. 2, pp. 55–82.

[37] Gwangmin Park, Daehyun Ku, Seonghun Lee, Woong-Jae Won e Wooyoung Jung. “Test methods of the AUTOSAR application software components”. Em: *International Congress Center - The Society of Instrument and Control Engineers (ICCAS-SICE 2009)*. Fukuoka, Japão, 2009, pp. 2601–2606.

[38] Kai Petersen, Robert Feldt, Shahid Mujtaba e Michael Mattsson. “Systematic Mapping Studies in Software Engineering”. Em: *International conference on Evaluation and Assessment in Software Engineering (EASE 2008)*. Bari, Itália, 2008, pp. 68–77.

[39] Hassan Reza e Emanuel Grant. “A method to test concurrent systems using architectural specification”. Em: *Journal of Supercomputing* 39.3 (2007), pp. 347–357.

[40] Hassan Reza e Suhas Lande. “Model Based Testing Using Software Architecture”. Em: *International Conference on Information Technology: New Generations (ITNG 2010)*. Las Vegas, Estados Unidos, 2010, pp. 188–193.

[41] Debra Richardson e Alexander Wolf. “Software testing at the architectural level”. Em: *International Software Architecture Workshop (ISAW’96)*. San Francisco, Estados Unidos, 1996, pp. 68–71.

[42] David S Rosenblum. “Challenges in Exploiting Architectural Models for Software Testing”. Em: *International Workshop on the Role of Software Architecture in Testing and Analysis (Rosatea 1998)*. Sicília, Italy, 1998, pp. 49–53.

[43] Nick Rozanski e Eóin Woods. *Software systems architecture: Working with stakeholders using viewpoints and perspectives*. 2ª ed. Addison–Wesley Professional, 2011, p. 704.

[44] Giuseppe Scollo e Silvia Zecchini. “Architectural Unit Testing”. Em: *Electronic Notes in Theoretical Computer Science* 111 (2005), pp. 27–52.

[45] Jon M. Siegel. *Model Driven Architecture (MDA) – MDA Guide rev. 2.0*. Rel. téc. ormsc/14-06-0. Object Management Group, 2014, p. 15.

[46] Wei-Tek Tsai, Jerry Gao, Xiao Wei e Yinong Chen. “Testability of software in service-oriented architecture”. Em: *Annual International Computer Software and Applications Conference (COMPSAC 2006)*. Washington, Estados Unidos, 2006, pp. 163–168.

[47] Erik van Veenendaal. *Test Maturity Model integration*. `http://www.tmmi.org/pdf/TMMi.Framework.pdf`. [Online, acessado em: 08/07/2015]. 2012.

[48] Fuqing Wang, Shuai Wang e Yindong Ji. “An automatic generation method of executable test case using model-driven architecture”. Em: *Proceedings of the 4th International Conference on Innovative Computing, Information and Control (ICICIC 2009)*. Kaohsiung, Taiwan, 2009, pp. 389–393.

[49] Jane Webster e Richard Thomas Watson. “Analyzing the Past to Prepare for the Future: Writing a Literature Review”. Em: *MIS Quarterly* 26.2 (2002), pp. 13–23.

[50] Liu Yang, Yafen Li e Pu Wang. “Design and implementation of automatic generation of test cases based on model driven architecture”. Em: *Proceedings of the 2nd International Conference on Information Technology and Computer Science (ITCS 2010)*. Kiev, Ucrânia, 2010, pp. 344–347.

[51] Junmin Ye, Wei Dong e Zhichang Qi. “A Method to Generate Embedded Real-Time System Test Suites Based on Software Architecture Specifications”. Em: *International Conference for Young Computer Scientists (ICYCS 2008)*. Hunan, China, 2008, pp. 2325–2329.

[52] Michal Young. “Testing Complex Architectural Conformance Relations”. Em: *International Workshop on the Role of Software Architecture in Testing and Analysis (Rosatea 1998)*. Sicília, Italy, 1998, pp. 42–45.

Apêndice A

Autores Identificados

Na Tabela A.1 apresenta-se a lista de autores dos 27 estudos incluídos. Vale ressaltar que todos os autores dos estudos foram considerados, não apenas os primeiros autores. Na primeira coluna são listados os 66 autores, na segunda, as universidades em que esses autores fazem parte e, por fim, na última coluna os países onde estão as universidades.

Tabela A.1: Autores, universidades e pais

Autor	Universidade	País
Bertolino, Antonia	Istituto di Scienza e Tecnologie dell'Informazione A.Faedo	Itália
Brito, Patrick	State University of Campinas	Brasil
Bucaioni, Alessio	Mälardalen University	Suécia
Bucchiarone, Antonio	Istituto di Scienza e Tecnologie dell'Informazione A.Faedo	Itália
Chao, Architect	Architects, Taiwan Chapter	Taiwan
De Lemos, Rogério	Computing Laboratory	Reino Unido
Dias, Marcio	University of California Irvine	Estados Unidos
Ding, Xuemei	Fujian Normal University	China
Dong, Wei	National University of Defense Technology	China
Gao, Jerry	Arizona State University	Estados Unidos
Gnesi, Stefania	Istituto di Scienza e Tecnologie dell'Informazione A.Faedo	Itália
Goltz, Ursula	Institute for Programming and Reactive Systems	Alemanha
Grant, Emanuel	University of North Dakota	Estados Unidos
Hao, Wu	School of Automation Science and Electrical Engineering	China
Harrold, Mary	The Ohio State Univ ersit y	Estados Unidos
Inverardi, Paola	Universita' di L'Aquila	Itália
Javed, A	The University of Queensland	Austrália
Ji, Yindong	Tsinghua University	China
Jin, Zhenyi	1761 Business Center Drive	Estados Unidos
JinSong, Yu	School of Automation Science and Electrical Engineering	China
Jung, Wooyoung	Daegu Gyeongbuk Institute of Science and Technolog	Coreia do Sul

Autor	Universidade	País
Kum, Daehyun	Daegu Gyeongbuk Institute of Science and Technology	Coreia do Sul
Lachmann, Remo	Institute of Software Engineering and Automotive Informatics	Alemanha
Lande, Suhas	University of North Dakota	Estados Unidos
Lee, Seonghun	Daegu Gyeongbuk Institute of Science and Technology	Coreia do Sul
LeiLi, Hu	National Defense Science and Technology Key Laboratory of Fire Control Technology	China
Leroux, Elena	University of South-Brittany	França
Li, Fu-Shiau	Brogent Technologies	Taiwan
Li, Yafen	Beijing University of Technology	China
Lity, Sascha	Institute for Programming and Reactive Systems	Alemanha
Lochau, Malte	Real-Time system Lab	Alemanha
Lun, Lijun	Harbin Normal University	China
Ma, Wei-Ming	Cheng Shiu University	Taiwan
Marinescu, Raluca	Mälardalen University	Suécia
Martins, Eliane	State University of Campinas	Brasil
Muccini, Henry	Universita' di L'Aquila	Itália
Offutt, Jeff	George Mason University	Estados Unidos
Oquendo, Flavio	University of South-Brittany	França
Park, Gwangmin	Daegu Gyeongbuk Institute of Science and Technology	Coreia do Sul
Pelliccione, Patrizio	Universita' di L'Aquila	Itália
Pettersson, Paul	Mälardalen University	Suécia
Pierini, Pierluigi	Universita' di L'Aquila	Itália
Qi, Zhichang	National University of Defense Technology	China
Reza, Hassan	University of North Dakota	Estados Unidos
Richardson, Debra	University of California Irvine	Estados Unidos
Rosenblum, David	University of California	Estados Unidos
Rubira, Cecília	State University of Campinas	Brasil
Saadatmand, Mehrdad	Mälardalen University	Suécia
Schaefer, Ina	Institute of Software Engineering and Automotive Informatics	Alemanha
Scollo, Giuseppe	University of Verona	Itália
Seceleanu, Cristina	Mälardalen University	Suécia
Strooper, P	The University of Queensland	Austrália
Tsai, Wei-Tek	Arizona State University	Estados Unidos
Wang, Fuqing	Tsinghua University	China
Wang, Pu	Beijing University of Technology	China
Wang, Shuai	Tsinghua University	China
Watson, G	The University of Queensland	Austrália
Wei, Xiao	San Jose State University	Estados Unidos

Autor	Universidade	País
Wolf, Alexander	University of Colorado	Estados Unidos
Won, Woong-Jae	Daegu Gyeongbuk Institute of Science and Technology	Coreia do Sul
Xiong, Qin	University of South-Brittany	França
Yang, Liu	Beijing University of Technology	China
Yichen, Wang	Beihang University	China
Young, Michal	University of Oregon	Estados Unidos
Zecchini, Silvia	University of Verona	Itália
ZhanBao, Gao	School of Automation Science and Electrical Engineering	China

Apêndice B

Strings de busca

Como descrito no Capítulo 2, a *string* de busca foi adaptada para cada máquina de busca. As máquinas SCOPUS, Web of Science e Science Direct possuem um bom mecanismo para busca por título, resumo e palavras-chave conjuntamente. Porém, a máquina ACM permite que sejam buscados por título, resumo e palavras-chave separadamente (por exemplo, caso uma parte da *string* esteja no título e a outra no resumo esse estudo não seria considerado como resultado). Assim, não foi possível realizar a busca na ACM utilizando apenas uma *string* e a busca na ACM foi realizada utilizando 9 *strings* de buscas. Essas *strings* foram criadas por meio de permutação entre as duas partes da *strings* e a adição das palavras *Abstract*, *Title* e *Keywords*. Por exemplo: considere as duas partes da *string*: A = {termo$_{a1}$ or termo$_{a1}$ or .. or termo$_{an}$} e B = {termo$_{b1}$ or termo$_{b1}$ or .. or termo$_{bn}$}; resultando na *string* "(A) and (B)". Aplicando o mecanismo, as *strings* geradas têm o seguinte formato:

- Abstract:(A) and Abstract:(B);
- Keywords:(A) and Keywords:(B);
- Title:(A) and Title:(B);
- Abstract:(A) and Title:(B);
- Abstract:(A) and Keyword:(B);
- Title:(A) and Keywords:(B);
- Title:(A) and Abstract:(B);
- Keywords:(A) and Abstract:(B);
- Keywords:(A) and Title:(B).

A máquina de busca IEEE possui a opção de busca nos metadados (titulo, resumo, palavras-chave definidas pelo autor e palavras-chave definidas pelo IEEE) do estudo. No entanto, utilizando o campo

palavras-chave definidas pela IEEE foram retornados muitos estudos não relevantes, uma vez que os termos adotados eram genéricos. Para contornar esta situação, realizou-se um procedimento análogo ao utilizado na ACM. Limitando as buscas apenas por título, resumo e palavras-chave definidas pelo autor.

B.1 Strings adaptadas às bases

As strings que foram utilizadas neste mapeamento serão apresentadas nesta seção.

B.1.1 Strings aplicadas à IEEE

As Figuras B.1 - B.9 contém as strings utilizadas na máquina de busca IEEE.

```
("Abstract": "Software Architecture"   OR   "Abstract": "Software Architectures")  AND
("Abstract": "Testing"                 OR
 "Abstract": "Testing Activity"        OR   "Abstract": "Testing Activities"       OR
 "Abstract": "Testing Criterion"       OR   "Abstract": "Testing Criteria"         OR
 "Abstract": "Test Plan"               OR   "Abstract": "Test Plans"               OR
 "Abstract": "Test Case"               OR   "Abstract": "Test Cases"               )
```

Figura B.1: Adaptação da string para IEEE: “Abstract” x “Abstract”)

```
("Author Keywords": "Software Architecture"   OR   "Author Keywords": "Software Architectures")  AND
("Abstract": "Testing"                        OR
 "Abstract": "Testing Activity"               OR   "Abstract": "Testing Activities"       OR
 "Abstract": "Testing Criterion"              OR   "Abstract": "Testing Criteria"         OR
 "Abstract": "Test Plan"                      OR   "Abstract": "Test Plans"               OR
 "Abstract": "Test Case"                      OR   "Abstract": "Test Cases"               )
```

Figura B.2: Adaptação da string para IEEE: “Author Keywords” x “Abstract”)

```
("Abstract": "Software Architecture"   OR   "Abstract": "Software Architectures")  AND
("Author Keywords": "Testing"          OR
 "Author Keywords": "Testing Activity" OR   "Author Keywords": "Testing Activities"  OR
 "Author Keywords": "Testing Criterion" OR  "Author Keywords": "Testing Criteria"    OR
 "Author Keywords": "Test Plan"        OR   "Author Keywords": "Test Plans"          OR
 "Author Keywords": "Test Case"        OR   "Author Keywords": "Test Cases"          )
```

Figura B.3: Adaptação da string para IEEE: “Abstract” x “‘Author Keywords”)

```
("Abstract": "Software Architecture"   OR   "Abstract": "Software Architectures")  AND
("Document Title": "Testing"           OR
 "Document Title": "Testing Activity"  OR   "Document Title": "Testing Activities"  OR
 "Document Title": "Testing Criterion" OR   "Document Title": "Testing Criteria"    OR
 "Document Title": "Test Plan"         OR   "Document Title": "Test Plans"          OR
 "Document Title": "Test Case"         OR   "Document Title": "Test Cases"          )
```

Figura B.4: Adaptação da string para IEEE: “Abstract” x “‘Document Title”)

("Document Title":	"Software Architecture"	OR	"Document Title":	"Software Architectures")	AND
("Abstract":	"Testing"	OR			
"Abstract":	"Testing Activity"	OR	"Abstract":	"Testing Activities"	OR
"Abstract":	"Testing Criterion"	OR	"Abstract":	"Testing Criteria"	OR
"Abstract":	"Test Plan"	OR	"Abstract":	"Test Plans"	OR
"Abstract":	"Test Case"	OR	"Abstract":	"Test Cases"	)

Figura B.5: Adaptação da string para IEEE: “Document Title” x “‘Abstract”)

("Author Keywords":	"Software Architecture"	OR	"Author Keywords":	"Software Architectures")	AND
("Author Keywords":	"Testing"	OR			
"Author Keywords":	"Testing Activity"	OR	"Author Keywords":	"Testing Activities"	OR
"Author Keywords":	"Testing Criterion"	OR	"Author Keywords":	"Testing Criteria"	OR
"Author Keywords":	"Test Plan"	OR	"Author Keywords":	"Test Plans"	OR
"Author Keywords":	"Test Case"	OR	"Author Keywords":	"Test Cases"	)

Figura B.6: Adaptação da string para IEEE: “Author Keywords” x “‘Author Keywords”)

("Document Title":	"Software Architecture"	OR	"Document Title":	"Software Architectures")	AND
("Document Title":	"Testing"	OR			
"Document Title":	"Testing Activity"	OR	"Document Title":	"Testing Activities"	OR
"Document Title":	"Testing Criterion"	OR	"Document Title":	"Testing Criteria"	OR
"Document Title":	"Test Plan"	OR	"Document Title":	"Test Plans"	OR
"Document Title":	"Test Case"	OR	"Document Title":	"Test Cases"	)

Figura B.7: Adaptação da string para IEEE: “Document Title” x “‘Document Title”)

("Document Title":	"Software Architecture"	OR	"Document Title":	"Software Architectures")	AND
("Author Keywords":	"Testing"	OR			
"Author Keywords":	"Testing Activity"	OR	"Author Keywords":	"Testing Activities"	OR
"Author Keywords":	"Testing Criterion"	OR	"Author Keywords":	"Testing Criteria"	OR
"Author Keywords":	"Test Plan"	OR	"Author Keywords":	"Test Plans"	OR
"Author Keywords":	"Test Case"	OR	"Author Keywords":	"Test Cases"	)

Figura B.8: Adaptação da string para IEEE: “Document Title” x “‘Author Keywords”)

("Author Keywords":	"Software Architecture"	OR	"Author Keywords":	"Software Architectures")	AND
("Document Title":	"Testing"	OR			
"Document Title":	"Testing Activity"	OR	"Document Title":	"Testing Activities"	OR
"Document Title":	"Testing Criterion"	OR	"Document Title":	"Testing Criteria"	OR
"Document Title":	"Test Plan"	OR	"Document Title":	"Test Plans"	OR
"Document Title":	"Test Case"	OR	"Document Title":	"Test Cases"	)

Figura B.9: Adaptação da string para IEEE: “Author Keywords” x “‘Document Title”)

B.1.2 Strings aplicadas à ACM

As Figuras B.10 - B.18 contém as strings utilizadas na máquina de busca ACM.

```
Abstract: ( "Software Architecture"   or  "Software Architectures")  and
Abstract: ( "Testing"                 or
            "Testing Activity"        or  "Testing Activities"       or
            "Testing Criterion"       or  "Testing Criteria"         or
            "Test Plan"               or  "Test Plans"               or
            "Test Case"               or  "Test Cases"               )
```

Figura B.10: Adaptação da string para ACM: “Abstract” x “Abstract”)

```
Abstract: ( "Software Architecture"   or  "Software Architectures")  and
Keywords: ( "Testing"                 or
            "Testing Activity"        or  "Testing Activities"       or
            "Testing Criterion"       or  "Testing Criteria"         or
            "Test Plan"               or  "Test Plans"               or
            "Test Case"               or  "Test Cases"               )
```

Figura B.11: Adaptação da string para ACM: “Abstract” x “Keywords”)

```
Keywords: ( "Software Architecture"   or  "Software Architectures")  and
Abstract: ( "Testing"                 or
            "Testing Activity"        or  "Testing Activities"       or
            "Testing Criterion"       or  "Testing Criteria"         or
            "Test Plan"               or  "Test Plans"               or
            "Test Case"               or  "Test Cases"               )
```

Figura B.12: Adaptação da string para ACM: “Keywords” x “Abstract”)

```
Abstract: ( "Software Architecture"   or  "Software Architectures")  and
   Title: ( "Testing"                 or
            "Testing Activity"        or  "Testing Activities"       or
            "Testing Criterion"       or  "Testing Criteria"         or
            "Test Plan"               or  "Test Plans"               or
            "Test Case"               or  "Test Cases"               )
```

Figura B.13: Adaptação da string para ACM: “Abstract” x “Title”)

```
   Title: ( "Software Architecture"   or  "Software Architectures")  and
Abstract: ( "Testing"                 or
            "Testing Activity"        or  "Testing Activities"       or
            "Testing Criterion"       or  "Testing Criteria"         or
            "Test Plan"               or  "Test Plans"               or
            "Test Case"               or  "Test Cases"               )
```

Figura B.14: Adaptação da string para ACM: “Title” x “Abstract”)

```
Keywords: ( "Software Architecture"   or  "Software Architectures")  and
Keywords: ( "Testing"                 or
            "Testing Activity"        or  "Testing Activities"       or
            "Testing Criterion"       or  "Testing Criteria"         or
            "Test Plan"               or  "Test Plans"               or
            "Test Case"               or  "Test Cases"               )
```

Figura B.15: Adaptação da string para ACM: “Keywords” x “Keywords”)

```
Title: ( "Software Architecture"   or  "Software Architectures")  and
Title: ( "Testing"                 or
         "Testing Activity"        or  "Testing Activities"       or
         "Testing Criterion"       or  "Testing Criteria"         or
         "Test Plan"               or  "Test Plans"               or
         "Test Case"               or  "Test Cases"               )
```

Figura B.16: Adaptação da string para ACM: “Title” x “Title”)

```
Title: ( "Software Architecture"   or  "Software Architectures")  and
Keywords: ( "Testing"              or
            "Testing Activity"     or  "Testing Activities"       or
            "Testing Criterion"    or  "Testing Criteria"         or
            "Test Plan"            or  "Test Plans"               or
            "Test Case"            or  "Test Cases"               )
```

Figura B.17: Adaptação da string para ACM: “Title” x “Keywords”)

```
Keywords: ( "Software Architecture"   or  "Software Architectures")  and
Title: ( "Testing"                    or
         "Testing Activity"           or  "Testing Activities"       or
         "Testing Criterion"          or  "Testing Criteria"         or
         "Test Plan"                  or  "Test Plans"               or
         "Test Case"                  or  "Test Cases"               )
```

Figura B.18: Adaptação da string para ACM: “Keywords” x “Title”)

B.1.3 String aplicadas à Science Direct

A Figura B.19 contém a string utilizada na máquina de busca Science Direct.

```
TITLE-ABSTR-KEY ( ( "Software Architecture"   OR  "Software Architectures")  AND
                  ( "Testing"                 OR
                    "Testing Activity"        OR  "Testing Activities"       OR
                    "Testing Criterion"       OR  "Testing Criteria"         OR
                    "Test Plan"               OR  "Test Plans"               OR
                    "Test Case"               OR  "Test Cases"               ) )
```

Figura B.19: Adaptação da string para Science Direct.

B.1.4 String aplicadas à Web of Science

A Figura B.20 contém a string utilizada na máquina de busca Web of Science.

```
Tópico: ( "Software Architecture"   OR  "Software Architectures")  AND
Tópico: ( "Testing"                 OR
          "Testing Activity"        OR  "Testing Activities"       OR
          "Testing Criterion"       OR  "Testing Criteria"         OR
          "Test Plan"               OR  "Test Plans"               OR
          "Test Case"               OR  "Test Cases"               )
```

Figura B.20: Adaptação da string para Web of Science.

B.1.5 String aplicada à SCOPUS

A Figura B.21 contém a string utilizada na máquina de busca SCOPUS.

```
TITLE-ABS-KEY ( (  {Software Architecture}   OR  {Software Architectures})  AND
                (  {Testing}                 OR
                   {Testing Activity}        OR  {Testing Activities}       OR
                   {Testing Criterion}       OR  {Testing Criteria}         OR
                   {Test Plan}               OR  {Test Plans}               OR
                   {Test Case}               OR  {Test Cases}               ) )
```

Figura B.21: Adaptação da string para SCOPUS.

Apêndice C

Formulário para extração de dados

Dados do estudo:

- Título do estudo
- Ano de publicação

Com relação a abordagem do estudo:

- Qual o objetivo do estudo?
- Como o autor descreve a arquitetura?
- Como o autor descreve o teste?
- Como o autor relaciona o teste e a arquitetura?
- São definidos:
  - ❑ Critérios
  - ❑ Técnicas
  - ❑ Métodos
  - ❑ Modelos
  - ❑ Outro:
- Qual fase de teste é abordada:
  - ❑ Unidade
  - ❑ Integração
  - ❑ Sistema
  - ❑ Regressão
  - ❑ Aceitação
  - ❑ Outro:

Com relação as informações arquiteturais

- O que tem sido utilizado?
  - ❑ ADL
  - ❑ Visões arquiteturais
  - ❑ Modelos específicos
  - ❑ Estilos arquiteturais
  - ❑ Decisões arquiteturais
  - ❑ Outro:

Com relação a ferramentas

- Nome da ferramenta
- Propósito da ferramenta
- Com relação a ferramenta
  - ❑ Ela foi avaliada
  - ❑ Ela é um protótipo
  - ❑ Ela está disponível

Com relação avaliação do estudo

- Tipo de avaliação
  - ❑ Estudo de caso
  - ❑ *Survey*
  - ❑ Experimento
  - ❑ Exemplo
  - ❑ Outro
- Onde o estudo foi conduzido
  - ❑ Academia
  - ❑ Indústria
  - ❑ Ambos
  - ❑ Outro:
- Tipo do estudo
  - ❑ Proposta
  - ❑ Editorial
  - ❑ Estudo Secundário
  - ❑ Outro:

Figura C.1: Formulário para extração de dados

Apêndice D

Veículos de Publicação Identificados

Nas Tabelas D.1 - D.4 são apresentados os veículos em que os estudos identificados (apresentados nas Tabelas 2.4 e 2.5) foram publicados. As colunas dessas tabelas representam: nome do veículo; quantidade de estudos (QE) publicados; porcentagem (PO) de estudos publicados em relação ao total de estudos inseridos; os estudos (E) inseridos que foram publicados nos veículo apresentados na coluna nome do veículo e, por fim, o meio de busca (B) utilizado para obter o estudo, que pode ser: (i) máquina de busca (M), (ii) *snowballing* (S) e (iii) indicação por especialista (E).

Tabela D.1: Lista de estudos publicados em conferência

	Nome	**QE**	**PO**	**E**	**B**
1	Conference on Information Technology and Computer Science	1	4%	S23	M
2	Computer Software and Applications Conference	1	4%	S12	M
3	Conference on Information Technology: New Generations	1	4%	S22	S
4	Conference for Young Computer Scientists	1	4%	S17	M
5	Conference on Innovative Computing, Information and Control	1	4%	S21	M
6	Conference on Intelligent Systems Design and Applications	1	4%	S16	M
7	Conference on Software Engineering and Advanced Applications	2	7%	S14,S27	M,S
8	ICROS-SICE International Joint Conference	1	4%	S20	M
9	Conference on the Quality of Software Architectures	1	4%	S09	M
10	Conference on Software Engineering Advances	1	4%	S25	E
11	Conference on Industrial Electronics and Applications	1	4%	S24	M
12	Conference on Secure Software Integration and Reliability Improvement	1	4%	S19	S
13	Conference on Fundamental Approaches to Software Engineering	1	4%	S07	M
	Total	14	52%	-	-

Tabela D.2: Lista de estudos publicados em simpósio

	Nome	**Q**	**P**	**E**	**B**
1	International Symposium on Software Reliability Engineering	1	4%	S05	M
	Total	1	4%	-	-

Tabela D.3: Lista de estudos publicados em revista

	Nome	**QE**	**PO**	**E**	**B**
1	IEEE Transactions on Software Engineering	1	4%	S08	M
2	Journal of Computer Science and Technology	1	4%	S18	M
3	Journal of Supercomputing	1	4%	S15	M
4	Journal of Systems and Software	2	7%	S11, S26	M
5	Electronic Notes in Theoretical Computer Science	1	4%	S10	S
	Total	6	22%	-	

Tabela D.4: Lista de estudos publicados em *workshop*

	Nome	**QE**	**PO**	**E**	**B**
1	International Software Architecture Workshop	1	4%	S01	M
2	International Workshop on the Role of Software Architecture in Testing and Analysis	3	11%	S02,S03,S04	S
3	Second International Workshop on Automation of Software Test	1	4%	S13	S
4	Applying Formal Methods: Testing, Performance, and M/E-Commerce: FORTE	1	4%	S06	M
	Total	6	22%	-	-