Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Formal Testing Approaches for Service-Oriented Architectures and Web Services: a Systematic Review

André Takeshi Endo
Adenilso da Silva Simão

Nº 348

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
Março/2010

Formal Testing Approaches for Service-Oriented Architectures and Web Services: a Systematic Review[*]

André Takeshi Endo
Adenilso da Silva Simão

Departamento de Sistemas de Computação
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo – Campus de São Carlos
Laboratório de Engenharia de Software
Caixa Postal 668, 13560-970 – São Carlos, SP, Brasil
e-mail: `{aendo,adenilso}@icmc.usp.br`

Abstract: Service-oriented architectures and web services have been used as important technologies to foster the development of loosely coupled and distributed applications. Web services pose new challenges for the testing activity, mainly because mission-critical and complex business process systems are implemented with them. In this context, formal testing approaches are necessary to guarantee the service quality. In this document, we identify formal approaches proposed to test service-oriented architectures and web services, analyzing the application context, models, empirical study, authorship, and references. A systematic review was planned and executed to select the studies considered in this work. We observed that, though more experiments and cooperation are necessary, there is a considerable interest in the topic.

Keywords: web services, service oriented architecture, software testing, formal testing, systematic review.

[*]**Acknowledgements:** The authors are thankful to Fabiano Cutigi Ferrari, Marcelo Medeiros Eler, and Marco Aurélio Graciotto Silva for providing useful comments and suggestions on this document. This work was financially supported by FAPESP/Brazil (process 2009/01486-9) and CNPq/Brazil (process 305236/2009-1).

Contents

1 Introduction **1**

1.1 Organization . . . 3

2 Planning and Conducting the Review **4**

2.1 Planning . . . 4

2.2 Conduction . . . 7

2.3 Threats to Validity . . . 10

2.4 Final Remarks . . . 11

3 Formal Testing Approaches for Web Services **12**

3.1 Description of Studies . . . 14

4 Analysis of Results **24**

4.1 Application Context . . . 24

4.1.1 Single Service Testing . . . 25

4.1.2 Service Composition Testing . . . 27

4.2 Models . . . 28

4.3 Empirical Study . . . 30

4.4 Authorship . . . 31

4.5 References Connection . . . 32

4.6 Final Remarks . . . 32

5 Conclusion and Future Work **33**

References **41**

A Data Extraction Form **42**

Chapter 1

Introduction

Service-Oriented Architecture (SOA) is an emergent architectural style that aims at creating loosely coupled software. SOA is based on standards and provides a distributed and protocol-independent computing paradigm. Software resources are encapsulated as *services* which are well-defined and self-contained modules with business functionalities, and being independent of state and context of other services (Papazoglou and Heuvel, 2007). Web services represent the most widespread technology that implements SOA. Web services are applications that provide operations invoked by other applications via Web, using open XML standards, such as SOAP (SOAP, 2003), WSDL (W3C, 2007), and UDDI (OASIS, 2005).

In a SOA environment, a set of services can be used to create new and more complex services that solve business process objectives. This process, called web service composition, can be developed either as orchestration or as choreography. In service orchestration, there is a main entity that is responsible for coordinating the partner services. The partner service is not aware of the composition. Currently, the main language to implement an orchestration is the Web Service Business Process Execution Language (BPEL) (Jordan et al., 2007). In service choreography, there is no control entity and all partner services work cooperatively. All services must be aware of the composition. There are several languages used to describe a service choreography, e.g. Web Services Choreography Description Language (WS-CDL) (Kavantzas et al., 2005) and Web Service Choreography Interface (WSCI).

As new software technologies, SOA and web services bring challenges for testing activity, such as dynamic binding, lack of observability, and service composition. In the last years, testing of web services and SOAs has been a intensively studied topic. Formal approaches have been applied to test services, as the importance of SOA-based software development demands that rigorous testing approaches are employed. Formal approaches use a mathematical model or specification, e.g., Finite State Machines (Lee and Yannakakis, 1996), Labeled Transition Systems (Tretmans, 1995), and control flow graphs (Rapps and Weyuker, 1985; Zhu et al., 1997), to support the test activity. According to Hierons et al. (2009), the presence of formal models or specifications can lead to more efficient and effective testing. New testing approaches are usually validated by an empirical study, assessing its effectiveness and efficiency. According to Briand (2007), empirical research is crucial to compare and improve software testing techniques and practices. In this context, there is an interest in answering questions such as "which are the formal approaches to test SOAs and web services?" and "how were these approaches evaluated?".

Trying to answer more generic questions about service testing, some surveys can be found in the literature (Bucchiarone et al., 2007; Canfora and Di Penta, 2009). Canfora and Di Penta (2009) report a survey of the main achievements related to SOA testing. The key challenges are identified, grouping the approaches found in the literature in test levels, namely like unit, integration, regression, and non-functional testing. The testability in SOA is also discussed. Bucchiarone et al. (2007) analyze the current proposals to test service composition. The studies are classified based on the two types of service composition: orchestration and choreography.

Although these surveys provide an overview of the service testing area, they were accomplished without a systematic process and do not handle the formal testing of web services in-depth. Seeking a methodology for performing rigorous review of the current research, the systematic review, original from the medicine, has been proposed for the software engineering community (Kitchenham and Charters, 2007). The systematic review follows a well defined and strict sequence of methodological steps defined in a previously established protocol (Biolchini et al., 2005). It is conceived to be an unbiased and (to a certain degree) repeatable process, summarizing the existing evidence related to a specific technology and providing a background to place new studies (Kitchenham and Charters, 2007).

In this document, we present a systematic review that shows studies in applying formal testing to SOAs and web services. The proposed approaches that use formal testing were selected, analyzing quantitatively the application context, models, empirical study, authorship, and references. We identified 32 adequate studies, showing the current interest in this topic. However, it was noted that more effort to empirical evaluation and cooperation among research groups are necessary. We believe that the results presented in this report are relevant for researcher seeking for an overview of the state-of-art of the formal testing of web services.

1.1 Organization

The technical report is organized as follows. Chapter 2 presents the planning and conduction of the systematic review. Chapter 3 shows a brief review of the selected studies. Chapter 4 shows the obtained results and an analysis of the selected studies. Finally, in Chapter 5, we present some conclusions and discuss future work.

Chapter

2

Planning and Conducting the Review

The systematic review technique has been proposed for the software engineering area to provide a more rigorous process to gather and evaluate evidence in a specific topic. It provides a systematic process to search, select, and analyze primary studies. Kitchenham and Charters (2007) define primary studies as individual studies that contribute to a systematic review. The systematic review is a secondary study since it reviews and synthesizes primary studies related to a research question.

We present the planning and conduction of the systematic review in the Sections 2.1 and 2.2, respectively. The threats to validity are presented in Section 2.3.

2.1 Planning

The systematic review is based on concepts and guidelines proposed by Kitchenham and Charters (2007) and Biolchini et al. (2005). The objective of this systematic review is to identify formal approaches to test SOAs and web services. Moreover, we also aim at identifying the application context, the adopted models, the type of evaluation, and the relation among research groups. Based on these goals, we formulate the following questions:

Main Research Question Which approaches have been proposed to test SOAs and web services using formal models or specifications?

Secondary Questions

1. Which are the application contexts, e.g., single service and service composition?
2. What formal models or specifications are used?
3. How are the proposed approaches for testing SOAs and web services evaluated?
4. What are the main research groups working in this topic?
5. What are the most referenced studies?

The primary study selection was carried out by searching in a set of the selected repositories. We used repositories that (1) have a search engine on the Web, (2) have a search mechanism able to use keywords, and (3) contain computer science papers. Our repositories include ACM Digital Library[1], IEEE Xplore Digital Library[2], SpringerLink[3], Scopus[4], and Compendex[5]. As electronic search procedures could not be sufficient, it is recommended the use of manual searches (see, e.g., (Jorgensen and Shepperd, 2007)). Thus, we also constructed a list of potential relevant studies based on manual searches.

We considered only primary studies written in English. Based on the research question and the background of the area, we defined the following keywords and synonyms:

- web services: SOA, service oriented, service based, service centric, service driven.
- testing: test.
- Formal: model, specification, mathematical.

Using these keywords and synonyms, we created the search string presented as follows. This string was adapted for each search engine since they present different mechanisms.

```
("web service" OR "web services" OR soa OR "service oriented" OR "service based" OR "service
centric" OR "service driven") AND (testing OR test) AND (formal OR model OR specification
OR mathematical)
```

[1]http://portal.acm.org/dl.cfm?coll=portal
[2]http://ieeexplore.ieee.org/
[3]http://www.springerlink.com/
[4]http://www.scopus.com/
[5]http://www.engineeringvillage.com/

Inclusion criteria were defined to evaluate whether or not a primary study will be selected for the systematic review. The inclusion criteria for this review are defined as follows:

- (A) The study must be available on the Web.
- (B) The study must have as its main focus to test SOAs or web services.
- (C) The study must be based on some formal model or specification.
- (D) The study must contain some type of evaluation like examples, case study, experiment, or proof of correctness. This evaluation must contain some kind of analysis, showing the achieved results.

We also planned the selection process, defining six steps necessary to carry out the systematic review, namely:

1. Searches in the repositories: Initially, strings were generated by means of the selected keywords. The searches were executed respecting the characteristics of each repository. The search was configured for the computer science subject and conference or journal papers in the repositories that provide this function. The constructed list of studies (manual search) was also included in this step.
2. Intermediate selection: an intermediate selection were carried out based on criteria (A) and (B). The title and the abstract were read for each primary study returned by the search engines.
3. Eliminate redundancies: there were some redundancies since a same primary study was returned by different search engines. In this step, we eliminated and recorded these redundancies.
4. Final selection: we applied the criteria (C) and (D) for the primary studies selected in the intermediate selection to obtain a list of studies. If, after reading the abstract, we could not determine whether the study should be selected, the primary study was read completely.
5. Eliminate divergences: If there are any divergences or doubts about the studies, a specialist could be consulted to solve them.

6. Analysis of references: Based on the selected primary studies, their related work references were analyzed and new studies were included if they satisfied the inclusion criteria. For each new selected study, we analyzed why the study was not selected in the searches. The reasons were documented and changes could be considered in a future replication of this systematic review.

Finally, each selected primary study was completely read and pieces of information were extracted based on a pre-defined form, designed to collect all the information needed to answer the research questions. As suggested by Kitchenham and Charters (2007), we defined this form during the planning to reduce the opportunity for bias. A filled form example is presented in Appendix A.

2.2 Conduction

The systematic review was conducted during four months (from September/2009 to December/2009). We applied the inclusion criteria incrementally. Figure 2.1 presents the selection process and preliminary numbers of selected studies in each step. Initially, we obtained a set of 104 primary studies using the criteria A and B. We noted that, even though our search string is focused on formal testing, we found 104 studies related to service testing. That is, a more generic string will match more studies in service testing.

Applying the criterion C (formal testing), we reduced this set to 61 studies. In the last step, the criterion D (evaluation) was applied and more studies were excluded. A set with 30 studies was selected considering all the inclusion criteria (A, B, C, and D) described in Section 2.1.

As we reported in the selection process, the analysis of references (step 6 in selection process) of the 30 studies added more 2 studies. We obtained and analyzed a final set of 32 primary studies. Table 2.1 presents a summary of returned studies, included studies, and the usage percentage for each repository. It is important to emphasize that there are some overlaps between different repositories. For instance, the five included studies found in Scopus were also returned by Compendex.

Based on the final set of 32 primary studies, we map the main publication characteristics related to year, type, and subject. Figure 2.2 presents the number of studies per year. Although there are studies in service testing (found by our review) published in 2002, the first study that satisfies our inclusion criteria was published in 2004. We note a growth during the period from 2006 to 2008. We cannot consider the year 2009 since the searches

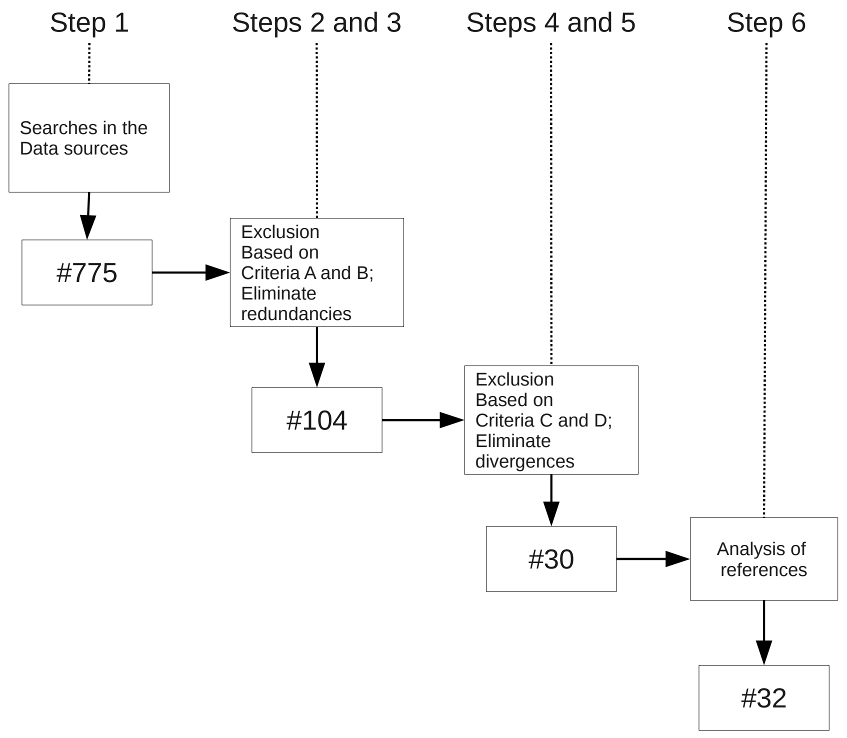

Step 1
Steps 2 and 3
Steps 4 and 5
Step 6
Searches in the Data sources
#775
Exclusion Based on Criteria A and B; Eliminate redundancies
#104
Exclusion Based on Criteria C and D; Eliminate divergences
#30
Analysis of references
#32

Figure 2.1: Selection process of primary studies.

were executed in September/2009 and relevant studies were likely not to be published or not yet indexed by the digital libraries.

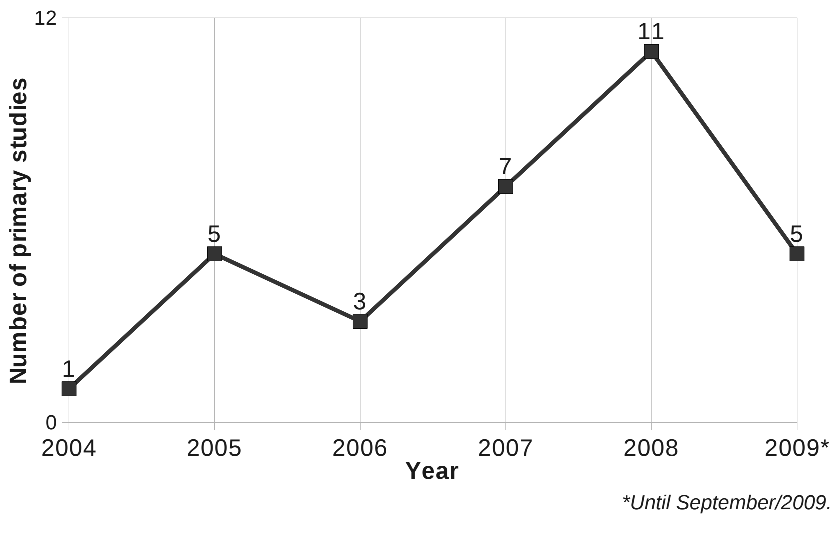


Figure 2.2: Number of primary studies per year.

We divide the publications into different forum types: workshops, conferences and symposiums, journals, and book chapters. This division is summarized in Figure 2.3. While book chapters, workshops, and journals present low percentages among the selected

Table 2.1: Returned and included studies for each source.

Repositories	**Returned**	**Included**	**%**
IEEE	201	8	3.98
ACM	146	7	4.79
Springer	78	8	10.26
Scopus	95	5	5.26
Compendex	233	16	6.87
Manual search	22	9	40.91
References	10	2	20.00

studies, conferences are the main forum type in formal testing of SOAs and web services (69%).

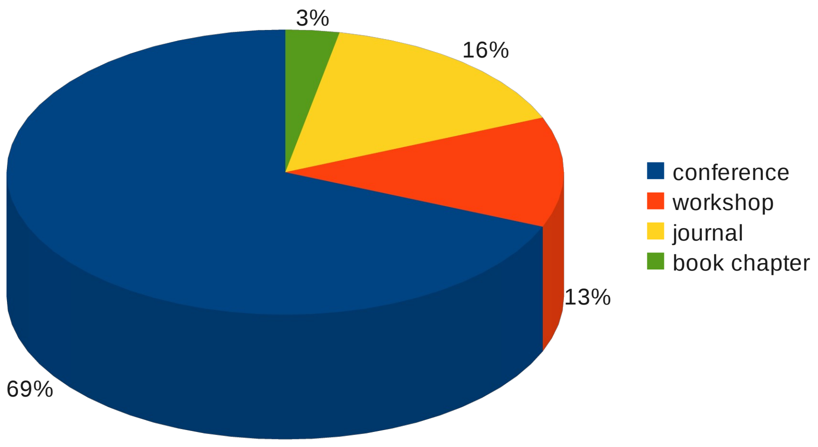


Figure 2.3: The percentage of primary studies per forum type.

Another characteristic considered was the subject of publication forum. We group the studies in three main subjects: (1) software engineering and testing, (2) SOA and Web technologies, and (3) others. Figure 2.4 shows the distribution considering these subjects. It is not a surprise that the subjects (1) and (2) concentrate 72% of studies. The subject (3) includes a reasonable number of studies since it includes conferences about general topics in computer science. Thus, researchers should be careful during their bibliography review conduction because considering only forums in subjects (1) and (2) can exclude relevant work.

Table 2.2 presents the publication forums and the number of studies for each one. There are more studies in COMPSAC and TESTCOM conferences. However, there is no concentration in determined forums since the majority of forums has just one selected study. This fact can cause problems for the researchers during an ad-hoc literature review because concentrating the searches on specific forums can exclude important studies.

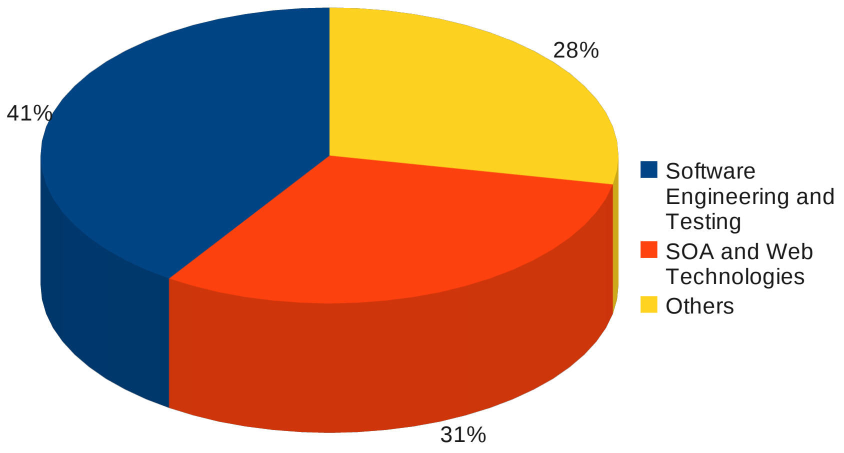
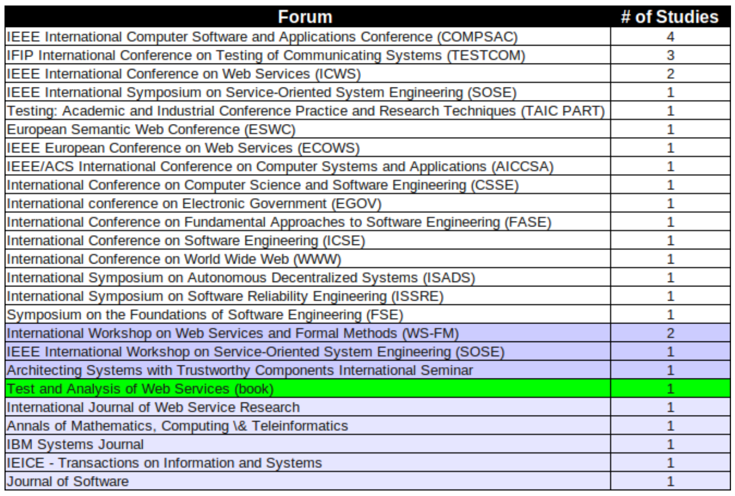


Figure 2.4: The percentage of primary studies per publication subject.

Table 2.2: Number of primary studies per forum.



Forum	# of Studies
IEEE International Computer Software and Applications Conference (COMPSAC)	4
IFIP International Conference on Testing of Communicating Systems (TESTCOM)	3
IEEE International Conference on Web Services (ICWS)	2
IEEE International Symposium on Service-Oriented System Engineering (SOSE)	1
Testing: Academic and Industrial Conference Practice and Research Techniques (TAIC PART)	1
European Semantic Web Conference (ESWC)	1
IEEE European Conference on Web Services (ECOWS)	1
IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)	1
International Conference on Computer Science and Software Engineering (CSSE)	1
International conference on Electronic Government (EGOV)	1
International Conference on Fundamental Approaches to Software Engineering (FASE)	1
International Conference on Software Engineering (ICSE)	1
International Conference on World Wide Web (WWW)	1
International Symposium on Autonomous Decentralized Systems (ISADS)	1
International Symposium on Software Reliability Engineering (ISSRE)	1
Symposium on the Foundations of Software Engineering (FSE)	1
International Workshop on Web Services and Formal Methods (WS-FM)	2
IEEE International Workshop on Service-Oriented System Engineering (SOSE)	1
Architecting Systems with Trustworthy Components International Seminar	1
Test and Analysis of Web Services (book)	1
International Journal of Web Service Research	1
Annals of Mathematics, Computing \& Teleinformatics	1
IBM Systems Journal	1
IEICE - Transactions on Information and Systems	1
Journal of Software	1

2.3 Threats to Validity

We identified some threats to validity of our systematic review. Two primary studies were not returned by the searches, but they were found during the analysis of references of the selected studies. These studies are indexed by the selected repositories, but we were not able to find them because the search string could not match them, since the abstracts do not contain the keyword or synonyms related to the word formal. We could not identify other synonyms that would improve the search string. Nevertheless, the inclusion of an additional step that analyzes the study references (step 6 of selection process) tends to overcome this threat.

The studies that do not satisfy the inclusion criterion (D) were excluded from our review. It is possible that some relevant work may have been excluded. However, we believe that extended versions, with consolidated work and more evaluation, will be published. Thus, these studies may appear in a future replication of this systematic review. This assumption is reasonable since we noted in our review that 11 selected studies are related to excluded studies.

2.4 Final Remarks

This chapter described the planning and conduction of the systematic review. The publication characteristics related to year, type, and subject were discussed considering the 32 selected studies. The threats to validity were also presented. The next chapter presents the 32 selected studies and a brief review for each of them.

Chapter 3

Formal Testing Approaches for Web Services

This chapter presents the 32 selected primary studies and a brief review for each one of them. Table 3.1 presents identification, title, authors, and year for each primary study. In Section 3.1, we annotate each paragraph with the study id. Thus, the reader can link the study information in Table 3.1 with its respective description in Section 3.1 and Figure 4.1 in Chapter 4.

Table 3.1: Information about the 32 selected primary studies.

Id	Title	Authors	Year
01	Developing and Assuring Trustworthy Web Services	Tsai, W.; Wei, X.; Chen, Y.; Xiao, B.; Paul, R.; Huang, H.	2005
02	Swiss Cheese Test Case Generation for Web Services Testing	Tsai, W.-T.; Wei, X.; Chen, Y.; Paul, R.; Xiao, B.	2005
03	A Robust Testing Framework for Verifying Web services by Completeness and Consistency Analysis	Tsai, W.; Wei, X.; Chen, Y.; Paul, R.	2005
04	A Metamorphic Testing Approach for Online Testing of Service-Oriented Software Applications	Chan, W.; Cheung, S.; Leung, K.	2007
05	WSDL-Based Automated Test Data Generation for Web Service	Ma, C.; Du, C.; Zhang, T.; Hu, F.; Cai, X.	2008

Id	Title	Authors	Year
06	Audition of Web Services for Testing Conformance to Open Specified Protocols	Bertolino, A.; Frantzen, L.; Polini, A.; Tretmans, J.	2004
07	Model-Based Generation of Testbeds for Web Services	Bertolino, A.; Angelis, G. D.; Frantzen, L.; Polini, A.	2008
08	On-The-Fly Model-Based Testing of Web Services with Jambition	Frantzen, L.; Las Nieves Huerta, M.; Kiss, Z. G.; Wallet, T.	2008
09	Control Flow Analysis and Coverage Driven Testing for Web Services	Li, L.; Chou, W.; Guo, W.	2008
10	Event-Driven Modeling and Testing of Web Services	Belli, F.; Linschulte, M.	2008
11	Generating Test Cases for Web Services Using Extended Finite State Machine	Keum, C.; Kang, S.; Ko, I.-Y.; Baik, J.; Choi, Y.-I.	2006
12	Formal Verification of Web Service Behavioural Conformance through Testing	Dranidis, D.; Kourtesis, D.; Ramollari, E.	2007
13	Leveraging Semantic Web Service Descriptions for Validation by Automated Functional Testing	Ramollari, E.; Kourtesis, D.; Dranidis, D.; Simons, A.	2009
14	Automated Functional Conformance Test Generation for Semantic Web Service	Paradkar, A.; Sinha, A.; Williams, C.; Johnson, R.; Outterson, S.; Shriver, C.; Liang, C.	2007
15	Automatic Conformance Testing of Web Services	Heckel, R.; Mariani, L.	2005
16	A Model-Driven Approach to Discovery, Testing and Monitoring of Web Services	Lohmann, M.; Mariani, L.; Heckel, R.	2007
17	Automatic Discovery of Web Services Based on Dynamic Black-Box Testing	Park, Y.; Jung, W.; Lee, B.; Wu, C.	2009
18	Towards the Testing of Composed Web Services in 3rd Generation Networks	Benharref, A.; Dssouli, R.; Glitho, R.; Serhani, M. A.	2006
19	Automatic Timed Test Case Generation for Web Services Composition	Lallali, M.; Zaidi, F.; Cavalli, A.; Hwang, I.	2008
20	BPEL4WS Unit Testing: Test Case Generation Using a Concurrent Path Analysis Approac	Yan, J.; Li, Z.; Yuan, Y.; Sun, W.; Zhang, J.	2006
21	An Automatic Test Case Generation Framework for Web Services	Zheng, Y.; Zhou, J.; Krause, P.	2007
22	Improving Trust in Composite eServices Via Run-Time Participants Testing	Corradini, F.; Angelis, F.; Polini, A.; Polzonetti, A.	2008
23	An Abstract Workflow-Based Framework for Testing Composed Web Services	Karam, M.; Safa, H.; Artail, H.	2007
24	A WS-BPEL Based Structural Testing Approach for Web Service Compositions	Liu, C.-H.; Chen, S.-L.; Li, X.-Y.	2008
25	Web Services Composition Testing: a Strategy Based on Structural Testing of Parallel Programs	Endo, A. T.; Simão, A. S.; Souza, S. R. S.; Souza, P. S. L.	2008

Id	Title	Authors	Year
26	Business-process-driven gray-box SOA testing	Li, Z. J.; Tan, H. F.; Liu, H. H.; Zhu, J.; Mitsumori, N. M.	2008
27	Data flow testing of service-oriented workflow applications	Mei, L.; Chan, W.; Tse, T.	2008
28	Test case prioritization for regression testing of service-oriented business applications	Mei, L.; Zhang, Z.; Chan, W. K.; Tse, T. H.	2009
29	An Empirical Study of the Use of Frankl-Weyuker Data Flow Testing Criteria to Test BPEL Web Services	Mei, L.; Chan, W. K.; Tse, T. H.; Kuo, F.-C.	2009
30	Verifying the conformance of web services to global interaction protocols: A first step	Baldoni, M.; Baroglio, C.; Martelli, A.; Patti, V.; Schifanella, C.	2005
31	Data flow testing of service choreography	Mei, L.; Chan, W. K.; Tse, T. H.	2009
32	Towards Automatic Regression Test Selection for Web Services	Ruth, M.; Oh, S.; Loup, A.; Horton, B.; Gallet, O.; Mata, M.; Tu, S.	2007

3.1 Description of Studies

Studies 01, 02 In Tsai et al. (2005b,c), the authors propose a technique to generate test cases for web service assurance and trustworthiness. Using a test case selection method called Swiss Cheese, test cases are generate to perform positive (required functionality) and negative (unspecified inputs or attacks) testing. The test process involves to convert an OWL-S specification into scenarios; extract boolean expressions from each scenario; and generate test cases selecting the most critical ones based on a model named Swiss Cheese. The Swiss Cheese map is an extended multi-dimensional Karnaugh map used to represent boolean expressions. The test case selection uses the criteria Hamming Distance (HD) and Boundary Count (BC) applied over the Swiss Cheese maps. HD and BC are measures used to evaluate the fault detection potency. The case study was carried to show the fault detection and validate the ranking strategy.

Study 03 Tsai et al. (2005a) propose a robust testing framework for web services. They propose an algorithm to identify completeness and consistency (CC). CC analysis is performed to identify incomplete and inconsistent conditions and constraints; categorize scenarios into verification and robustness patterns; generate positive and negative test cases using a pattern-oriented approach; and use Swiss Cheese model to select the most critical test cases. The approach is applied in the Flood Watching web service.

Study 04 Chan et al. (2007) propose a metamorphic approach for online service testing. Metamorphic testing is an approach to deal with the oracle problem using well-defined relations of multiple input-output pairs. It uses an initial successful test set to generate a new follow-up test set using these relations. They use the test cases for offline testing as initial test set. A metamorphic service is used as an access wrapper and implements the relations. In the online testing, it is validated if a service can interact correctly with other services using the follow-up test set. An experiment was carried out with a service-oriented calculator. It contains 16 classes and 2,480 lines of code. They used combinatorial testing to generate the initial test set. After that, they created faulty versions to evaluate the fault detection.

Study 05 Ma et al. (2008) propose an automated test data generation approach for single operations of web services. An input element model (tree with constraints) is constructed through the WSDL and XML schema files. The test data is generated in a bottom-up way, using black-box strategies as boundary value, equivalence class, and random testing. Algorithms for model building and data generation are also presented. Using the order request example, 4096 test data were generated and applied in three different versions.

Study 06 Bertolino et al. (2004) propose to augment the service description (WSDL) with Protocol State Machine (PSM). PSM is a diagram of UML 2.0 used to specify states, operation order, and associate pre- and post-conditions. The authors previously introduced an audition framework for automatic web service testing during the service registration. The PSM model is mapped to Symbolic Transition System (STS) for automatic test case generation. Using the **ioco**-conformance relation (Tretmans, 1996), test cases are generated on-the-fly.

Study 07 Bertolino et al. (2008) propose a model based approach to generate stubs for web services. The stubs are generated following Service Level Agreement (SLA) contracts and functional contracts (State Machines). The STS notation was used to specify the right values for invocation parameters, the right order of the messages and which are the features of a message provided by the service. A supporting tool, named PUPPET, generates a stub for the service using as input the STS model, the WS-Agreement specification, and the WSDL file. The generated stubs are able to simulate the functional and extra-functional behavior of the service. The authors consider two QoS properties: latency and reliability (maximal number of failures in a given time window). The work is focused on the Customer-Supplier-Warehouse case study and on the PUPPET support tool. The case study verifies the generated stubs and the detection of functional and extra-functional failures.

Study 08 Frantzen et al. (2008) present a tool, called Jambition, that generates on-the-fly test cases for web services. The work is part of the PLASTIC project[1]. The model STS or Service State Machine (SSM) is used to specify the functional aspects of the service. A service operation is related to a transition in the STS. There are three types of transitions: input (a message sent to the service), output (a message sent from the service), and unobservable. Variables and guard conditions can also be included. The testing approach implemented is random and on-the-fly. The input data is generated based on constraints (guard transitions) using the constraint solver of GNU prolog. This data is executed and the tool selects randomly a new operation on-the-fly. If some output is not allowed by the STS, the tool stops and reports a failure. The approach was evaluated using the Alarm Dispatcher service (eHealth). Errors were detected during the importation (the STS is checked with the service) and the validation. In the validation, errors related to guard condition violation, never ending operations, and missing features were also found. Jambition generates more test cases (35) than manual testing (5) to achieve full transition coverage.

Study 09 Li et al. (2008a) propose three methods to discover Control Flow Graphs (CFGs) of web services for analysis, verification, and testing. In this work, the CFG is defined as all legitimate sequences of successful web service operations, focusing on stateful and conversational web services. First, pre-conditions and effects are used to specify service states using specifications in Resource Description Framework (RDF) (W3C, 2004b). Second, RDF graphs are used to determine control flow relations between operations. Third, the authors maximize the test coverage with minimal test cases using the Google's PageRank algorithm (Langville and Meyer, 2006). Service operations are ranked based on the number visited operations before reach them and Markov Chain are used to model the linkage among the operations.

Study 10 Belli and Linschulte (2008) propose to use the *Event Sequence Graph* (ESG) notation to support web service testing. The requests and responses of the service are

[1]http://www.ist-plastic.org/

modeled as events. In positive testing, the legal event sequences are covered by complete event sequences generated by Chinese Postman Problem algorithm (Aho et al., 1995). The faulty events (negative testing) are covered using the deep-first search. Events with input data are refined through Decision Tables (DTs) that represent pre- and post-conditions, defining a contract. A case study with a hotel service is also presented, showing that more faults were found using negative testing.

Study 11 Keum et al. (2006) propose to use Extended Finite State Machines (EFSMs) for modeling web services. It is defined a procedure to derive an EFSM by means of a WSDL specification. The procedure is based on forms filling and no tool is provided. The algorithm proposed by Bourhfir et al. (1997) is used to generate test cases, covering control flow and data flow of the model. A banking service is used to illustrate the procedure and a telecommunication service is considered to carry out the experiments.

Study 12 Dranidis et al. (2007) introduce a new approach to verify the conformance between the service implementation and a formal behavior specification. The Stream X-Machine (SXM) is used to model the service behavior and its testing method is used to generate test cases. The transitions in a SXM are labeled with processing functions. XMDL-O language is used to describe the processing functions, modeling the request, response, pre-conditions, and effects. Applying the SXM testing method, a complete test set of input sequences can be generated to verify the implementation.

Study 13 Ramollari et al. (2009) propose a method based on semantic description of Inputs, Outputs, Preconditions and Effects (IOPE) to generate a SXM model. The method is focused on web services that maintain information about the internal state after invocations of operations (stateful). OWL is used to specify the service data model and RIF-PRD (Rule Interchange Format - Production Rule Dialect) for specifying the service behavior. RIF-PRD is applied to model the pre-conditions and effects for each service operation. The OWL ontology and the production rules in RIF-PRD are linked to the WSDL through SAWSDL. Using IOPE-descriptions, the service behavior is transformed into a SXM model, identifying states, inputs, outputs, transitions, merged states, and merged transitions. The memory, guard conditions, and memory updates are also identified. The SXM method generates test cases initially applying the W-method and converting function sequences into input sequences using a fundamental test function. A set of tools is used to transform and run JUnit test cases.

Study 14 Paradkar et al. (2007) present an approach for conformance testing through test case generation based on the IOPE paradigm. The authors classify the testing of semantic web services into 3 stages: development testing, repository testing, and end user testing. Test goals are derived for each operation including boundary values, cardinality constraints for collections, and the notion of fault sensitization. These test goals and service IOPE description are used as input for the planner component (also a suite of constraint solvers). Verification sequences are generated based on mutant states, e.g., no instance creation, no attribute update, and normal effects in exception. To produce executable test cases, a template is provided to add code fragments (necessary input and output). A case study was carried out, comparing the approach with manual testing. Comparable results in coverage and fault-detection were achieved, but the proposed approach saved substantial effort.

Study 15 In Heckel and Mariani (2005), the authors propose a high quality service registry that incorporates automated web service testing before the registration. Graph transformation rules (GT rules) are used to specify the behavior of the web service. GT rules specify the service behavior at the conceptual level (not the implementation level). The testing conformance is defined in terms of completeness (inside the input domain) and soundness (outside the input domain). The test case generation uses a domain-based strategy, named partition testing. The input domain is divided into subsets based on WSDL types and constraints of the GT rules. The def-use criterion is used to test data flow, interpreting the creation of nodes and edges as definitions and deletion as uses. Dependency relations are defined for generating test cases to cover the execution of pairs of rules.

Study 16 Lohmann et al. (2007) present a framework to guarantee high-quality service-oriented applications. The approach is based on testing, discovery, and monitoring techniques using GT rules and Java modeling language (JML) contracts. The service registration includes a testing phase where the registry automatically generates, executes, and evaluates test cases. Partition testing is used to test single operations and relations among transformation rules to test sequence of operations. The service discovery is based on matching the rules of the client and providers. Monitoring is proposed to verify the services at run-time. GT rules are transformed into JML constructs, enabling a model-driven monitoring. If the contracts are violated, exceptions are launched. The monitoring approach is proposed for the provider, but it can be adapted for the client with some modifications.

Study 17 Park et al. (2009) propose an automatic testing-based approach to discover web services. Delta-Grammar (GT rules) is used to describe the pre-conditions and effects for each service operation, similarly to work of Heckel and Mariani (2005). The discovery process initiates with the client sending a request with semantic information using GT rules. Keywords are extracted from the rules and searched in the WSDL repository. A simple signature matching is presented to map a request and a GT rule. Abstract test cases are randomly generated using the productions and refined for the execution. The test cases are executed concurrently for each selected candidate service. The test results are logged with web service data, test case information, and QoS attributes.

Study 18 Benharref et al. (2006) propose a multi-observer architecture to detect and locate faults in composed web services. The proposed architecture is composed of a global observer and local observers that cooperate to collect and manage faults found in the composed service. The observers can detect faults related to orchestration violation, state and timing faults, and input or output datatype differences. the concept of passive testing or passive observation is used, that is based on collecting and analyzing traces. Passive testing is composed by two steps: passive homing and fault detection. The homing procedure is applied to bring the observer to the same state as the observed service. In fault detection, each event (request or response) is checked against the behavioral model.

Study 19 Lallali et al. (2008) propose an approach to transform BPEL specification into an Intermediate Format (IF) to test timing constraints. The approach relies on the coverage of specific test purposes. IF model is based on the timed automaton extended with variables, communication primitives, and urgency attributes. Transition urgency is used to control the time progress. There are three types of attributes: (i) eager: is urgent and blocks the time progress; (ii) lazy: is never urgent and never blocks time progress; and (iii) delayable: allows waiting as long as time progress does not disable it. A tool called BPEL2IF is used to execute the transformation. A conformance relation is defined. A timed test case is also defined as an observable timed trace that validates some timed requirements. For the test case generation, the strategy Hit-or-Jump is adapted. The Hit condition specifies when a state satisfying one or more test purposes is reached. The Jump condition specifies when a depth limit is reached without satisfying a test purpose and a leaf node is selected to restart the search. This strategy is implemented in the TestGen-IF tool. The timed test purposes are defined using action, state, or clock constraints.

Study 20 Yan et al. (2006) propose a new method to generate test cases for BPEL programs. Initially, the method transforms BPEL programs into an *Extended Control Flow Graph* (XCFG) and generates all the sequential paths of the XCFG. These paths are combined in concurrent test paths. After that, the constraint solver BoNuS is used to solve the path restrictions and generate executable test cases. Some methods are presented to reduce the number of test cases, such as model exclusive edges and pairs. Two test coverage criteria are used: basis path coverage and user directed path coverage. Variable sharing is treated pre-defining an order or enumerating all possible orders.

Study 21 Zheng et al. (2007) propose a model called Web Service Automata (WSA) to represent the operational semantics of the BPEL. The WSA is a finite state machine with signature, data structures, and message storage schema. As WSA has no hierarchy, the hierarchical dependencies are modeled using parent and child relationships among machines. Considering fault and interruption, the machines for structured activities have a common layout of states and transitions. A structure called linkWrapper is used to handle links. The data dependencies are captured by three models: internal (a single BPEL process), external (one process to another process), and global (union of internal and external). Control-flow and data-flow structural criteria are encoded using the temporal logics *Linear Time Temporal Logic* (LTL) and *Computation Tree Logic* (CTL). Using the WSA, inputs for the model checkers SPIN and NuSMV are generated. The model checking application for test case generation is based on the idea of recovering test cases by means of counter-examples. In order to avoid the state space explosion and improve the performance, some model simplifications are introduced. A study case with a loan process example and a support tool are also presented.

Study 22 Corradini et al. (2008) propose a testing approach for run-time service composition. The idea is to reduce risks associated with a possible failure during a run-time composition. The orchestration just asks for information related to the services that need to be discovered. The discovered services are executed with a trial test suite. If some threat is detected, a message of unavailable service is sent to the client. The BPEL specification is translated into a BIR model (Robby et al., 2003). The BIR model is used as input for the BOGOR model checker. Using model checking and genetic algorithms, a set of traces are generated to cover interactions (with partner services). Each trace is projected with its relative data for test purposes. An example of a payment of fines for traffic violations is also presented.

Study 23 Karam et al. (2007) propose the Structural based Workflow Framework (SBWF) for testing web service composition. Besides handling the control and data flow, the semantic flow is also considered representing how services are choreographed to perform a request. For each type of flow, a type of graph is defined: Semantic Flow Graph (SFG), Control Flow Graph (CFG), and Data Flow Graph (DFG). The composition is modeled by these three types of graphs. The definition of test case is extended to include the nodes that must be executed. The All-States and All-Transitions are defined as test adequacy criteria for the proposed model.

Study 24 Liu et al. (2008) propose structural approach for testing BPEL programs. A test model called BCFG (WS-BPEL Control Flow Graph) is proposed to model the execution flow of a BPEL program. A set of different nodes and edges are defined based on BPMN notation. Specific solutions are given for links a scope structures. Using the BCFG, test paths are generated skipping the loops or exploring only once (includes node and edge coverage). After that, there is a step to analyze the test path conditions and remove the unfeasible paths. The loan process (with links) is used as example, generating 16 test paths. Among them, only 3 test paths are feasible.

Study 25 Endo et al. (2008) propose a test strategy to test BPEL based service composition. A test model for parallel programs named Parallel Control Flow Graph (PCFG) (Souza et al., 2008) is adapted for representing the BPEL structures. The test model focuses mainly on the communication by means of the send and receive primitives. New types of definition and use are proposed and test criteria are defined for the associations. The concept of required element groups is proposed to increase the accuracy of the testing criteria. The architecture of the supporting tool called ValiBPEL-Web is also presented. The strategy is applied on three examples showing results about the applicability.

Study 26 Li et al. (2008b) propose a gray-box approach to test business process specified in BPEL. The approach involves coverage for composition, regression testing for loose-coupling SOA, and test generation. To reach this goal, the authors use three key enablers: test-path exploration, trace analysis, and regression testing selection. In test path exploration, the BPEL Flow Graph (BFG) is used to model the BPEL, representing the BPEL structures with specific nodes. Concurrent test paths are generated and constraint solvers are used to generate input data. Coverage goals are also proposed including all-path coverage and branch coverage. In trace analysis, traces are collected during the testing and are mapped to a forest model (based on BPEL) to find a path. In regression testing selection, test cases generated for both old and new versions of a BPEL process are compared. A minimization algorithm is proposed to select the minimal test set that covers all impacted activities. A testing tool named BPELTester was implemented to support the approach. They carried out a preliminary case study with 12 real-world BPEL processes.

Study 27 Mei et al. (2008) propose coverage criteria for BPEL based on the X-WSBPEL model. X-WSBPEL is composed by a control flow graph to represent the BPEL program and XRGs (XPath Rewriting Graph) to represent the XPath expressions. The authors propose the existence of conceptual variables in the XRGs that are not directly represented as program variables. An algorithm for generating an XRG based on the XPath query and XML schema is also presented. New types of definition and use are proposed and testing criteria are defined based on them. Basically, the criteria aim to execute the new def-use associations proposed for the XRG. Their experimental results show that the approach detects more than 90% of the seeded faults. Faults were classified into in-BPEL, in-XPath, and in-WSDL.

Study 28 Mei et al. (2009c) propose a multilevel coverage model to capture the coverage of BPEL, XPath and WSDL artifacts. This model is used to support the regression testing and a family of test case prioritization techniques is also proposed. Their goal is to reorder a test suite according to the coverage data. The multilevel coverage model includes: Level 1 - CM-1 (workflow), level 2 - CM-2 (workflow, XPath), and level 3 - CM-3 (workflow, XPath, WSDL). CM-1 mainly involves the coverage of workflow branches; CM-2 involves the coverage of XRG branches; and CM-3 includes the coverage of WSDL elements. Two strategies are used to handle multiple types of artifacts: summation (treat different artifacts homogenously) and refinement (use another artifact only if the artifact already referred to cannot help). The experiment results show that the proposed techniques significantly outperform conventional techniques and confirm that to consider artifacts level by level is an effective strategy.

Study 29 Mei et al. (2009b) present an empirical study that applies the all-uses criterion to BPEL programs. The authors propose that BPEL testing has three levels: BPEL level (business logic), WSDL level (messages based on WSDL specification), and Web Service Level (integration BPEL and the web services). They work on the WSDL level, defining a mapping from BPEL to Control Flow Graph (CFG). The empirical study considered 8 BPEL programs and 60 faults were seeded. The faults were partitioned into 3 categories: in-BPEL, in-XPath, and in-WSDL. Using test sets with 100% all-uses coverage, the fault-detection rate was measured. The results show that the all-uses criterion is better to reveal faults in the BPEL artifacts than in WSDL and XPath artifacts.

Study 30 Baldoni et al. (2005) propose a framework for conformance testing in service choreographies, inspired by multi-agents systems. The authors focus on the problem of conformance between the peer behaviors and the global behavior (choreography). Both the peers and the choreography are modeled as finite state automata. A new automaton is built based on peer and global automata to verify the conformance and interoperability.

Study 31 Mei et al. (2009a) propose a new model to describe a service choreography that manipulates data flow by means of XPath queries. In a choreography, XPath queries can handle different XML schema files. The authors motivate the testing challenge so that XPath queries can raise different expectations which lead to faults in the choreography. The notion of XRG pattern is defined to represent valid messages for data manipulation. Based on LTS, the LTS-based Choreography model (C-LTS) is proposed with XRGs attached in transitions that represent service invocations. New types of def-use associations are proposed based on XRGs and XRG patterns and test adequacy criteria are presented. A case study was carried out with a real application, showing that the technique using XRG patterns can detect more than 90% of the seeded faults.

Study 32 Ruth et al. (2007) propose a framework for safe regression test selection in web services. The mechanism is based on the coverage information and the set of dangerous edges. A dangerous edge corresponds to program entities that can behave differently under a single test case due to differences between the original and the modified version. CFGs are generated for each single service and a global CFG is generated by merging the CFGs that are parts of a composition. Each CFG node has some fields that help the identification of changes. A call graph is used to track the potentially impacted operations and to order the regression testing process. A publish/subscribe mechanism is proposed to support the actors involved in the service testing. The authors present a case study with five services, showing the selectivity of the proposed regression testing approach.

Chapter

4

Analysis of Results

We analyzed the 32 selected primary studies considering the topics: application context, models, empirical study, authorship, and references connection. The topics were defined to answer the research questions presented in Section 2.1. Figure 4.1 summarizes the considered topics in a graph, modeling each primary study as a node (the node label is its id presented in Table 3.1). Node forms, edges, and colors represent the different topics considered. More explanation about Figure 4.1 and these topics are presented in the following sections.

4.1 Application Context

During the development of service based applications, the stakeholders involved in testing activity must handle different levels or application contexts. As in traditional software testing, it is common that different strategies are carried out for unit and integration testing. Analyzing the selected studies, there are two clearly distinct contexts: (1) single service testing and (2) service composition testing. Single service testing is basically interested in assuring the reliability of an isolated service. Service composition testing focuses on the integration and execution of a set of services contained in a composition.

Figure 4.1 represents single service studies as circles and service composition studies as squared boxes. We also present in this section the references for the 32 primary studies

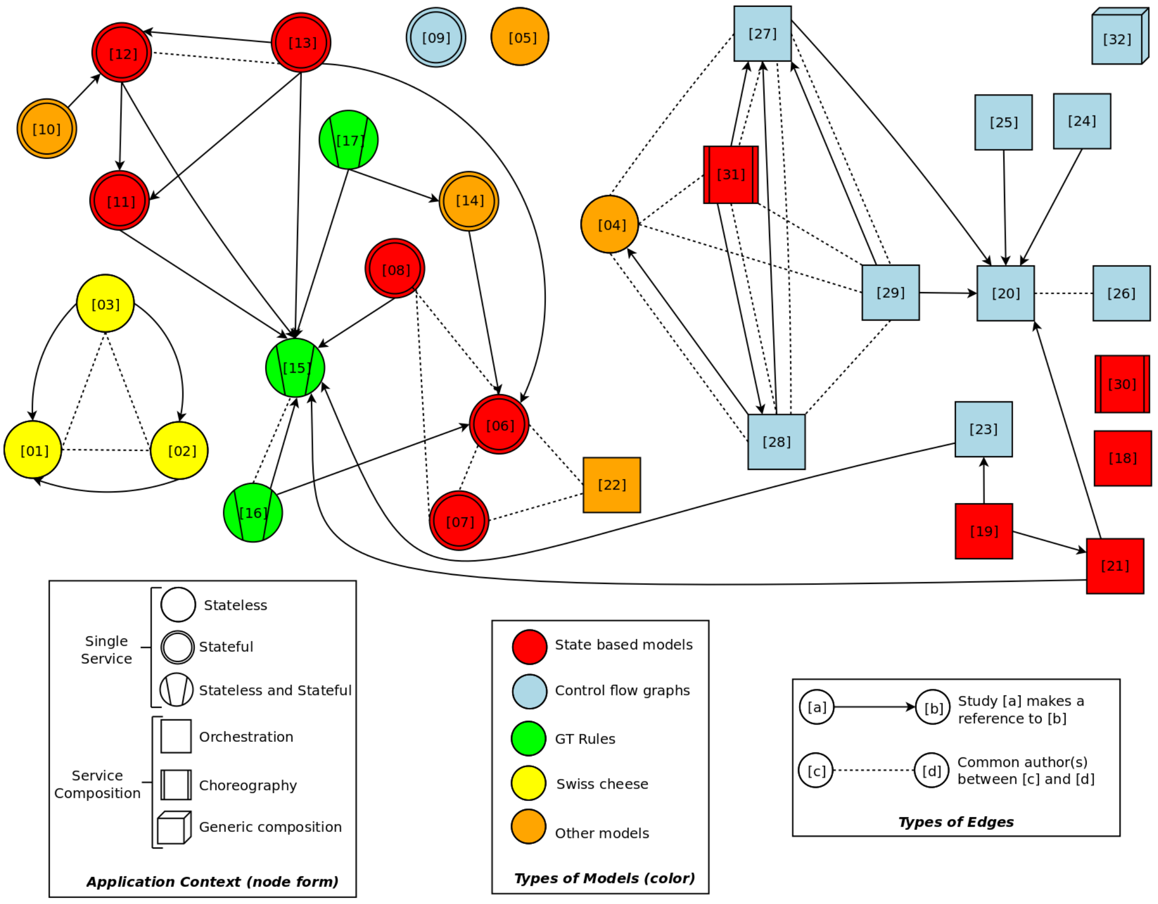


Figure 4.1: Relationships among the selected primary studies.

grouped in each application context. Table 4.1 presents the distribution of primary studies for each context. The references are also presented.

There is research interest in both application contexts with an almost equal distribution of studies (17 single vs 15 composition). We also note that there is no approach that fits for both single services and service composition since the studies focus on a specific context. It is clear that the current approaches can be combined to provide a more complete testing strategy. However, we could not identify any research effort in this context.

4.1.1 Single Service Testing

In the single service testing, the proposed approaches can be classified based on service state, i.e., whether a service keeps internal state or not. We consider a service stateless when it is not necessary a context (previous operation invocations) to invoke a service operation. Using a stateless service, only the input for the invocation is necessary. On the other hand, a stateful service keeps state information and the current state can change the

Table 4.1: Distribution of studies considering application context.

Type	References	# of Studies
Single Service Testing	(Tsai et al., 2005b), (Tsai et al., 2005c), (Tsai et al., 2005a), (Chan et al., 2007), (Ma et al., 2008), (Bertolino et al., 2004), (Bertolino et al., 2008), (Frantzen et al., 2008), (Li et al., 2008a), (Belli and Linschulte, 2008), (Keum et al., 2006), (Dranidis et al., 2007), (Ramollari et al., 2009), (Paradkar et al., 2007), (Heckel and Mariani, 2005), (Lohmann et al., 2007), (Park et al., 2009)	17
Service Composition Testing	(Benharref et al., 2006), (Lallali et al., 2008), (Yan et al., 2006), (Zheng et al., 2007), (Corradini et al., 2008), (Karam et al., 2007), (Liu et al., 2008), (Endo et al., 2008), (Li et al., 2008b), (Mei et al., 2008), (Mei et al., 2009c), (Mei et al., 2009b), (Baldoni et al., 2005), (Mei et al., 2009a), (Ruth et al., 2007)	15

output of a service operation. Table 4.2 presents the number of studies that are applicable for stateless, stateful, and both types of services. The references for each context are also presented. Figure 4.1 presents different circles for each context of single services.

Table 4.2: Distribution of single service testing.

Applied For	References	# of Studies
Stateless Service	(Tsai et al., 2005b), (Tsai et al., 2005c), (Tsai et al., 2005a), (Chan et al., 2007), (Ma et al., 2008)	5
Stateful Service	(Bertolino et al., 2004), (Bertolino et al., 2008), (Frantzen et al., 2008), (Li et al., 2008a), (Belli and Linschulte, 2008), (Keum et al., 2006), (Dranidis et al., 2007), (Ramollari et al., 2009), (Paradkar et al., 2007)	9
Stateless and Stateful	(Heckel and Mariani, 2005), (Lohmann et al., 2007), (Park et al., 2009)	3

In *stateless services*, the studies focus on test case generation. The work of Tsai et al. (2005a,b,c) is based on the Swiss Cheese model derived from semantic specifications in Ontology Web Language for Service (OWL-S) (W3C, 2004a). Ma et al. (2008) use available WSDL and XML Schema specifications to support the tests. Using a different approach, Chan et al. (2007) use metamorphic relations to deal with the oracle problem and generate new test cases.

In *stateful services*, conformance testing is the main goal using test case generation to support it (Belli and Linschulte, 2008; Bertolino et al., 2004; Dranidis et al., 2007; Frantzen et al., 2008; Keum et al., 2006; Paradkar et al., 2007; Ramollari et al., 2009). Bertolino et al. (2008) focus on stub generation using a state model and QoS properties. Li et al. (2008a) propose the usage of control flow graphs to model the valid sequence of operation invocations.

In *stateless and stateful services*, all studies are based on Graph Transformation rules, enabling the applicability for stateless and stateful services (Heckel and Mariani, 2005; Lohmann et al., 2007; Park et al., 2009). Besides supporting test case generation, the process of service discovery is also supported.

We note more interest in stateful services (9 studies) than stateless services (5 studies). A group of studies focuses on automating the process of test case generation mainly as consequence of the dynamic nature of web services. Some studies consider that the syntactic specification (WSDL) was augmented with semantic descriptions (OWL-S, RDF) (Ramollari et al., 2009; Tsai et al., 2005b). There are few approaches that can be applied to stateless and stateful services (only 3 studies).

4.1.2 Service Composition Testing

Service composition testing is divided based on the used paradigms: orchestration and choreography. Table 4.3 presents the number of studies that can be applied for orchestration, choreography, and both paradigms (generic). The references for each paradigm are also presented. Figure 4.1 presents different squared boxes for each context of service composition.

Table 4.3: Distribution of service composition testing.

Applied For	**References**	**# of Studies**
Orchestration	(Benharref et al., 2006), (Lallali et al., 2008), (Yan et al., 2006), (Zheng et al., 2007), (Corradini et al., 2008), (Karam et al., 2007), (Liu et al., 2008), (Endo et al., 2008), (Li et al., 2008b), (Mei et al., 2008), (Mei et al., 2009c), (Mei et al., 2009b)	12
Choreography	(Baldoni et al., 2005), (Mei et al., 2009a)	2
Generic	(Ruth et al., 2007)	1

In *orchestration testing*, BPEL is the main language used to describe the composition (10 studies). There is research focused on test case generation handling different aspects, such as BPEL specific structures (Liu et al., 2008; Zheng et al., 2007), concurrency (Li et al., 2008b; Yan et al., 2006), timing properties (Lallali et al., 2008), and run-time composition (Corradini et al., 2008). There are also studies related to coverage criteria that: use workflow modeling (Karam et al., 2007), handle communication among processes (Endo et al., 2008), consider XPath artifacts (Mei et al., 2008), and evaluate traditional data flow criteria (Mei et al., 2009b). Regression testing is also approached considering minimization (Li et al., 2008b) and prioritization (Mei et al., 2009c). Moreover, Benharref et al. (2006) propose a observer architecture to support passive testing.

In *choreography testing*, we identified 2 studies, one in conformance testing (Baldoni et al., 2005) and other in coverage criteria (Mei et al., 2009a). Baldoni et al. (2005) propose a framework inspired in multi-agent systems for conformance testing between the peer (single service) behavior and the global behavior (choreography). Mei et al. (2009a) propose some test adequacy criteria for service choreography specified in WS-CDL that manipulates XPath queries.

We identified only one study that handles composition in a *generic* way, i.e., there is no distinction between orchestration and choreography. Ruth et al. (2007) propose an approach for safe regression testing through the usage of control flow graphs to identify the changes.

There is a large difference between the number of studies based on orchestration (12 studies) and on choreography (2 studies). The composition languages cited in the analyzed work are BPEL (orchestration) and WS-CDL (choreography). A high number of studies in orchestration testing can be consequence of the maturity level of the composition languages. BPEL has been accepted as a standard language for orchestration and business process. In contrast, there is no consensus on a choreography language. Currently, WS-CDL is the most cited one. Only the work of Ruth et al. (2007) addresses the service composition in a generic way.

4.2 Models

In this section, we discuss the main models used to support the formal testing of web services. Table 4.4 shows this classification, presenting the references and the number of primary studies for each group. Figure 4.1 presents different colors for each group of models.

Table 4.4: Groups of adopted models.

Group	**References**	**# of Studies**
State Based Models	(Baldoni et al., 2005; Benharref et al., 2006; Bertolino et al., 2008, 2004; Dranidis et al., 2007; Frantzen et al., 2008; Keum et al., 2006; Lallali et al., 2008; Mei et al., 2009a; Ramollari et al., 2009; Zheng et al., 2007)	11
Control Flow Graphs	(Endo et al., 2008; Karam et al., 2007; Li et al., 2008a,b; Liu et al., 2008; Mei et al., 2008, 2009b,c; Ruth et al., 2007; Yan et al., 2006)	10
Graph Transformation Rules	(Heckel and Mariani, 2005; Lohmann et al., 2007; Park et al., 2009)	3
Swiss Chess Model	(Tsai et al., 2005a,b,c)	3
Other Models	(Belli and Linschulte, 2008; Chan et al., 2007; Corradini et al., 2008; Ma et al., 2008; Paradkar et al., 2007)	5

State Based Models In this group, the studies use state based models to support the test activity, ranging from simple to complex models. These models are Finite State Machine (FSM) and extension (Benharref et al., 2006; Keum et al., 2006), Labeled Transition System (LTS) and extension (Bertolino et al., 2008, 2004; Frantzen et al., 2008; Mei et al., 2009a), Stream X-Machine (SXM) (Dranidis et al., 2007; Ramollari et al., 2009), finite state automaton (Baldoni et al., 2005), timed automaton (Lallali et al., 2008), and Web Service Automaton (WSA) (Zheng et al., 2007). The models are used to describe the control flow, data flow, and timing properties. While most of the studies reuse or adapt established models, Zheng et al. (2007) propose the WSA to represent the semantic operation of the BPEL language.

Control Flow Graph and Extensions Control Flow Graph (CFG) and well-known extensions like Def-Use Graph (Rapps and Weyuker, 1985) have constantly been used for supporting structural testing. In the selected set, CFGs and extensions are used mainly to test BPEL based service composition (Endo et al., 2008; Karam et al., 2007; Li et al., 2008b; Liu et al., 2008; Mei et al., 2008, 2009b; Yan et al., 2006). However, some studies do not use CFGs to support structural testing. In Li et al. (2008a), CFGs model the sequences of operations for testing stateful services. Ruth et al. (2007) and Mei et al. (2009c) apply CFGs and coverage criteria to support regression testing.

Graph Transformation Rules Graph Transformation (GT) rules are used to augment the service specification with pre-conditions, effects, and a notion of states. A GT rule refines the service operation, adding information about the parameters and internal data. The service state can be recorded as a graph attribute, though GT rules is not a state based model. As a consequence, GT rules based approaches are adequate for stateless and stateful services, supporting discovery, monitoring, automatic testing, and regression testing (Heckel and Mariani, 2005; Lohmann et al., 2007; Park et al., 2009).

Swiss Chess Model Swiss chess model is the basis for the work of Tsai et al. (2005a,b,c). It is a model based on boolean expressions, extracted from semantic specifications (OWL-S), that are represented by Karnaugh maps and finally a Swiss Cheese map. The model is used to generate positive and negative test cases.

Other Models We identified 5 primary studies which use models that cannot be classified into the previous groups. Table 4.5 summarizes these models and the application context.

Table 4.5: Studies classified as *other models*.

Reference	Model	Application Context
(Chan et al., 2007)	Metamorphic relations	Stateless services
(Ma et al., 2008)	input element type model (tree)	Stateless services
(Belli and Linschulte, 2008)	Event Sequence Graph	Stateful services
(Paradkar et al., 2007)	IOPE	Stateful services
(Corradini et al., 2008)	BIR model (checking)	Service composition

Remarks Each primary study addresses the model construction in a specific form. There are two common approaches: (1) to suppose that there exists a formal model and (2) to generate a model based on service specifications. The first approach is commoner in single service testing due to the fact that only syntactic specifications (WSDL, XML Schema) are available. The second approach is most used in composition testing, since supposing that there are BPEL or WS-CDL specifications in this context is reasonable. Semantic specifications are also considered in this way, though they are not frequently available.

Despite the classification of models, we note two issues related to the selected studies: non-functional properties and model checking. We found studies that handle non-functional properties related to Service Level Agreement (SLA) and timing properties. SLA plays an important role in the interaction between a provider and a client. The testing activity can be very helpful to verify SLA properties like latency and reliability (Bertolino et al., 2008). In this context, the timing related properties were also considered (Benharref et al., 2006; Lallali et al., 2008).

Model checking has been used to verify whether a model meets pre-defined properties related to the software behavior. We found some studies that use model checking to support test case generation, reusing developed model checking tools (Corradini et al., 2008; Lallali et al., 2008; Zheng et al., 2007).

4.3 Empirical Study

According to Do et al. (2005), determining the type of empirical study carried out in the studies requires some degree of subjective judgment, since the description and quantitative measures can be unclear. Based on previously proposed guidelines, Do et al. (2005) classify the empirical studies performed in the software testing area into: controlled experiments, case studies, and examples. We use this classification to divide the primary studies selected by the systematic review. Each type of empirical study is defined as follows (Do et al., 2005):

- Controlled Experiment: a study that manipulates factors such as number of programs, number of versions (faulty versions), faults, and test cases.
- Case Study: a study that cannot be classified as an experiment but contains non-trivial or real-world samples.
- Example: a study that uses only single trivial samples.

Considering this classification, we analyzed the set of primary studies. Figure 4.2 presents the percentage for each type of empirical study. We note that a high number of studies were classified as *example* (47%) and *case study* (34%). It implies that a low number of *controlled experiments* (19%) has been carried out. This fact indicates that more empirical research, mainly controlled experiments, should be performed to evaluate the proposed approaches for formal testing of web services.

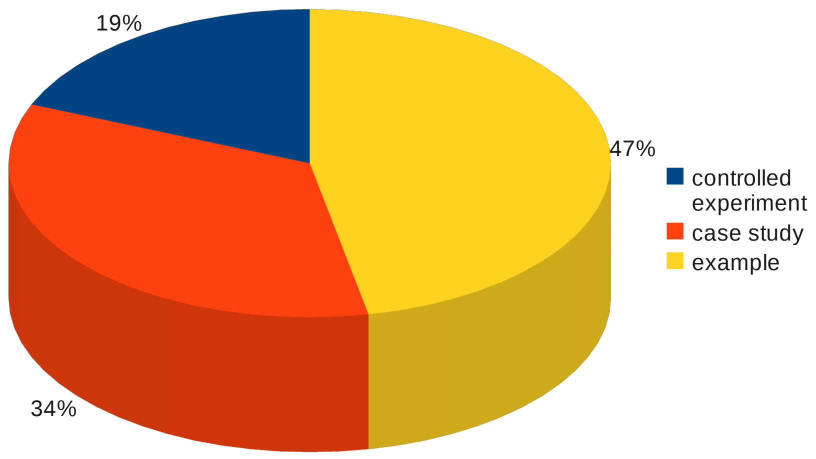


Figure 4.2: The percentage of empirical studies.

Among the selected set, only one primary study reports a quantitative comparison with an equivalent approach (Keum et al., 2006). We believe that two main facts hinder the execution of comparative empirical studies: benchmarks and tools. First, the primary studies report empirical evaluation with their own programs and configuration. This fact hampers possible comparison among the proposed approaches. It would be interesting to establish a common benchmark containing a set of services and necessary artifacts to overcome this limitation. Second, the existence of supporting tools would facilitate the execution of experiments, making this activity less manual and error-prone. We note that a low number of tools is reported and, moreover, only two tools (Jambition (Frantzen et al., 2008) and PUPPET (Bertolino et al., 2008))[1] were found available for download on the Internet.

4.4 Authorship

In Figure 4.1, dotted edges connect studies with at least one common author. Based on this information, we identified two main research groups working on formal testing for web services. The first group (5 studies) works with service composition testing (orchestration and choreography). The second group (4 studies) concentrates their effort on testing of

[1]http://plastic.isti.cnr.it/wiki/tools

stateful services. We also note another small groups (3 or 2 studies) and a large number of isolated studies.

Note in Figure 4.1 that there is no connection among the groups which implies that no cooperation have been carried out in this topic. This fact can justify the lack of approaches or strategies that handle testing of both single services and compositions.

4.5 References Connection

In Figure 4.1, the references were represented as directed edges. An directed edge from node A to node B means that the primary study A makes a reference to the study B. The two more referenced studies are coincidentally one in single service testing (Heckel and Mariani (2005) - 8 references) and one in service composition testing (Yan et al. (2006) - 5 references). Another important fact is the low connection among the studies. The primary study that makes more references to other studies is connected with 4 studies (Ramollari et al. (2009)). Moreover, there are many primary studies with zero or one connection.

The low reference connection suggests the presence of problems to find related studies and the lack of secondary studies that summarize the topic.

4.6 Final Remarks

This chapter presented an analysis of different topics concerning the selected primary studies. We quantified the studies considering the application context, models, and empirical study. The authorship and references connection were also discussed to view the research groups and cooperation. In the next chapter, we present the conclusions of this work, as well as the possibilities of future work.

Chapter

5

Conclusion and Future Work

In this report, we presented a systematic review on formal testing for SOAs and web services. A set of 32 primary studies was selected based on searches in the main repositories for the computer science area. We noted an increasing interest in formal testing for web services. A mapping of the publication forums was also presented.

The selected studies focus on testing aspects for single services and service compositions. We grouped and discussed about the formal models used to support the tests. The empirical evaluation carried out in the studies was classified based on the work of Do et al. (2005). The authorship and references connection related to the studies were also discussed.

We conclude that there is a considerable amount of research on formal testing for web services, though some points deserve more attention. One point is that more effort is necessary to evaluate the proposed approaches, conducting controlled experiments, making comparisons, and evaluating fault detection with more details. In this context, an important issue is the availability of benchmarks and tools to support empirical studies. Based on the analysis of authorship and references, we noted a low level of interaction among the research groups. We identified many types of adopted models, providing a good range of options for the service developers. However, this range of options cannot be found in choreography testing with only 2 selected studies.

As future work, we plan the following activities:

- Replicate the systematic review to update the selected work, evolving the planning and analysis and reducing the threats to validity.

- Use this systematic review and published surveys as a starting point to produce a complete systematic mapping (such as in Petersen et al. (2008)) of the service testing area.

References

Aho, A. V.; Dahbura, A. T.; Lee, D.; Uyar, M. U. An optimization technique for protocol conformance test generation based on uio sequences and rural chinese postman tours, p. 427–438. 1995.

Baldoni, M.; Baroglio, C.; Martelli, A.; Patti, V.; Schifanella, C. Verifying the conformance of web services to global interaction protocols: A first step. In: *International Workshop on Web Services and Formal Methods (WS-FM)*, Springer, 2005, p. 257–271.

Belli, F.; Linschulte, M. Event-driven modeling and testing of web services. In: *IEEE International Computer Software and Applications Conference (COMPSAC)*, 2008, p. 1168–1173.

Benharref, A.; Dssouli, R.; Glitho, R.; Serhani, M. A. Towards the testing of composed web services in 3rd generation networks. In: *IFIP International Conference on Testing of Communicating Systems (TESTCOM)*, Springer Berlin / Heidelberg, 2006, p. 118–133.

Bertolino, A.; Angelis, G. D.; Frantzen, L.; Polini, A. Model-based generation of testbeds for web services. In: *IFIP International Conference on Testing of Communicating Systems (TESTCOM)*, Tokyo, Japan, 2008, p. 266–282 (*Lecture Notes in Computer Science*, v.5047).

Bertolino, A.; Frantzen, L.; Polini, A.; Tretmans, J. Audition of web services for testing conformance to open specified protocols. In: *Architecting Systems with Trustworthy Components International Seminar*, 2004, p. 1–25.

Biolchini, J.; Mian, P. G.; Natali, A. C. C.; Travassos, G. H. *Systematic review in software engineering.* Relatório Técnico, Systems Engineering and Computer Science Department - COPPE/UFRJ, 2005.

Bourhfir, C.; Dssouli, R.; E.Aboulhamid; N.Rico Automatic executable test case generation for EFSM specified protocols. In: *International Workshop on Testing of Communicating Systems (IWTCS'97)*, 1997, p. 75–90.

Briand, L. C. A critical analysis of empirical research in software testing. In: *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Washington, DC, USA: IEEE Computer Society, 2007, p. 1–8.

Bucchiarone, A.; Melgratti, H.; Severoni, F. Testing service composition. In: *8th Argentine Symposium on Software Engineering (ASSE'07)*, Mar del Plata, Argentina, 2007.

Canfora, G.; Di Penta, M. Service-oriented architectures testing: A survey. In: *Software Engineering: International Summer Schools (ISSSE)*, Berlin, Heidelberg: Springer-Verlag, 2009, p. 78–105.

Chan, W.; Cheung, S.; Leung, K. A metamorphic testing approach for online testing of service-oriented software applications. *International Journal of Web Service Research*, v. 4, n. 2, p. 61–81, 2007.

Corradini, F.; Angelis, F.; Polini, A.; Polzonetti, A. Improving trust in composite eservices via run-time participants testing. In: *International conference on Electronic Government (EGOV)*, Berlin, Heidelberg: Springer-Verlag, 2008, p. 279–290.

Do, H.; Elbaum, S.; Rothermel, G. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering*, v. 10, n. 4, p. 405–435, 2005.

Dranidis, D.; Kourtesis, D.; Ramollari, E. Formal verification of web service behavioural conformance through testing. *Annals of Mathematics, Computing & Teleinformatics*, v. 1, n. 5, p. 36–43, 2007.

Endo, A. T.; Simão, A. S.; Souza, S. R. S.; Souza, P. S. L. Web services composition testing: a strategy based on structural testing of parallel programs. In: *Testing: Academic and Industrial Conference Practice and Research Techniques (TAIC PART)*, Windsor, United Kingdom, 2008, p. 3–12.

Frantzen, L.; Las Nieves Huerta, M.; Kiss, Z. G.; Wallet, T. On-the-fly model-based testing of web services with jambition. In: *International Workshop on Web Services and Formal Methods (WS-FM)*, Berlin, Heidelberg: Springer-Verlag, 2008, p. 143–157.

Heckel, R.; Mariani, L. Automatic conformance testing of web services. In: *International Conference on Fundamental Approaches to Software Engineering (FASE)*, Lecture Notes in Computer Science, 2005, p. 34–48 (*Lecture Notes in Computer Science*,).

Hierons, R. M.; Bogdanov, K.; Bowen, J. P.; Cleaveland, R.; Derrick, J.; Dick, J.; Gheorghe, M.; Harman, M.; Kapoor, K.; Krause, P.; Lüttgen, G.; Simons, A. J. H.; Vilkomir, S.; Woodward, M. R.; Zedan, H. Using formal specifications to support testing. *ACM Computing Surveys (CSUR)*, v. 41, n. 2, p. 1–76, 2009.

Jordan, D.; Evdemon, J.; Alves, A.; Arkin, A.; Askary, S.; Barreto, C.; Bloch, B.; Curbera, F.; Ford, M.; Goland, Y.; GuÃzar, A.; Kartha, N.; Liu, C. K.; Khalaf, R.; Konig, D.; Marin, M.; Mehta, V.; Thatte, S.; van der Rijn, D.; Yendluri, P.; Yiu, A. OASIS web services business process execution language (WSBPEL) v2.0. 2007.
Available at `http://docs.oasis-open.org/wsbpel/2.0/`

Jorgensen, M.; Shepperd, M. A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering*, v. 33, n. 1, p. 33–53, 2007.

Karam, M.; Safa, H.; Artail, H. An abstract workflow-based framework for testing composed web services. In: *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, 2007, p. 901–908.

Kavantzas, N.; Burdett, D.; Ritzinger, G.; Fletcher, T.; Lafon, Y.; Barreto, C. Web services choreography description language version 1.0. 2005.
Available at `http://www.w3.org/TR/ws-cdl-10/`

Keum, C.; Kang, S.; Ko, I.-Y.; Baik, J.; Choi, Y.-I. Generating test cases for web services using extended finite state machine. In: *IFIP International Conference on Testing of Communicating Systems (TESTCOM)*, Lecture Notes in Computer Science, New York, NY, USA: Springer, 2006, p. 103–117 (*Lecture Notes in Computer Science*,).

Kitchenham, B.; Charters, S. *Guidelines for performing systematic literature reviews in software engineering.* Relatório Técnico EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

Lallali, M.; Zaidi, F.; Cavalli, A.; Hwang, I. Automatic timed test case generation for web services composition. In: *IEEE European Conference on Web Services (ECOWS)*, 2008, p. 53–62.

Langville, A. N.; Meyer, C. D. *Google's PageRank and beyond: The science of search engine rankings.* Princeton, NJ: Princeton University Press, 2006.

Lee, D.; Yannakakis, M. Principles and methods of testing finite state machines - a survey. *Proceedings of the IEEE*, v. 84, n. 8, p. 1090–1123, 1996.

Li, L.; Chou, W.; Guo, W. Control flow analysis and coverage driven testing for web services. In: *IEEE International Conference on Web Services (ICWS)*, 2008a, p. 473–480.

Li, Z. J.; Tan, H. F.; Liu, H. H.; Zhu, J.; Mitsumori, N. M. Business-process-driven gray-box soa testing. *IBM Systems Journal*, v. 47, n. 3, p. 457–472, 2008b.

Liu, C.-H.; Chen, S.-L.; Li, X.-Y. A ws-bpel based structural testing approach for web service compositions. In: *IEEE International Symposium on Service-Oriented System Engineering (SOSE)*, 2008, p. 135–141.

Lohmann, M.; Mariani, L.; Heckel, R. A model-driven approach to discovery, testing and monitoring of web services. In: *Test and Analysis of Web Services*, 2007, p. 173–204.

Ma, C.; Du, C.; Zhang, T.; Hu, F.; Cai, X. WSDL-based automated test data generation for web service. In: *International Conference on Computer Science and Software Engineering (CSSE)*, 2008, p. 731–737.

Mei, L.; Chan, W.; Tse, T. Data flow testing of service-oriented workflow applications. In: *International Conference on Software Engineering (ICSE)*, New York, NY, USA: ACM, 2008, p. 371–380.

Mei, L.; Chan, W. K.; Tse, T. H. Data flow testing of service choreography. In: *Symposium on the Foundations of Software Engineering (FSE)*, New York, NY, USA: ACM, 2009a, p. 151–160.

Mei, L.; Chan, W. K.; Tse, T. H.; Kuo, F.-C. An empirical study of the use of frankl-weyuker data flow testing criteria to test bpel web services. In: *IEEE International Computer Software and Applications Conference (COMPSAC)*, 2009b, p. 81–88.

Mei, L.; Zhang, Z.; Chan, W. K.; Tse, T. H. Test case prioritization for regression testing of service-oriented business applications. In: *International Conference on World Wide Web (WWW)*, New York, NY, USA: ACM, 2009c, p. 901–910.

OASIS UDDI specifications tc. 2005.
Available at http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

Papazoglou, M. P.; Heuvel, W.-J. Service oriented architectures: approaches, technologies and research issues. *The International Journal on Very Large Databases (VLDB)*, v. 16, n. 3, p. 389–415, 2007.

Paradkar, A.; Sinha, A.; Williams, C.; Johnson, R.; Outterson, S.; Shriver, C.; Liang, C. Automated functional conformance test generation for semantic web services. In: *IEEE International Conference on Web Services (ICWS)*, 2007, p. 110–117.

Park, Y.; Jung, W.; Lee, B.; Wu, C. Automatic discovery of web services based on dynamic black-box testing. In: *IEEE International Computer Software and Applications Conference (COMPSAC)*, 2009, p. 107–114.

Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. Systematic mapping studies in software engineering. In: *International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2008.

Ramollari, E.; Kourtesis, D.; Dranidis, D.; Simons, A. Leveraging semantic web service descriptions for validation by automated functional testing. In: *European Semantic Web Conference (ESWC)*, 2009, p. 593–607.

Rapps, S.; Weyuker, E. J. Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, v. 11, n. 4, p. 367–375, 1985.

Robby; Dwyer, M. B.; Hatcliff, J. Bogor: an extensible and highly-modular software model checking framework. *ACM SIGSOFT Software Engineering Notes*, v. 28, n. 5, p. 267–276, 2003.

Ruth, M.; Oh, S.; Loup, A.; Horton, B.; Gallet, O.; Mata, M.; Tu, S. Towards automatic regression test selection for web services. In: *IEEE International Computer Software and Applications Conference (COMPSAC)*, 2007, p. 729–736.

SOAP SOAP specifications. 2003.
Available at http://www.w3.org/TR/soap/

Souza, S. R. S.; Vergilio, S. R.; Souza, P. S. L.; Simão, A. S.; Hausen, A. C. Structural testing criteria for message-passing parallel programs. *Concurrency and Computation: Practice and Experience*, v. 20, n. 16, p. 1893–1916, 2008.

Tretmans, G. J. Testing labelled transition systems with inputs and outputs. In: Cavalli, A.; Budkowski, S., eds. *Participants Proceedings of the Int. Workshop on Protocol Test Systems VIII — COST 247 Session, Evry, France*, Evry, France: Institut National des Télécommunications, 1995, p. 461–476.

Tretmans, J. Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, v. 17, n. 3, p. 103–120, 1996.

Tsai, W.; Wei, X.; Chen, Y.; Paul, R. A robust testing framework for verifying web services by completeness and consistency analysis. In: *IEEE International Symposium on Service-Oriented System Engineering (SOSE)*, 2005a, p. 151–158.

Tsai, W.; Wei, X.; Chen, Y.; Xiao, B.; Paul, R.; Huang, H. Developing and assuring trustworthy web services. In: *International Symposium on Autonomous Decentralized Systems (ISADS)*, 2005b, p. 43–50.

Tsai, W.-T.; Wei, X.; Chen, Y.; Paul, R.; Xiao, B. Swiss cheese test case generation for web services testing. *IEICE - Transactions on Information and Systems*, v. E88-D, n. 12, p. 2691–2698, 2005c.

W3C OWL-S: Semantic markup for web services. 2004a.
Available at `http://www.w3.org/Submission/OWL-S`

W3C RDF/XML syntax specification (revised). 2004b.
Available at `http://www.w3.org/TR/REC-rdf-syntax/`

W3C Web services description language (WSDL) version 2.0. 2007.
Available at `http://www.w3.org/TR/wsdl20`

Yan, J.; Li, Z.; Yuan, Y.; Sun, W.; Zhang, J. BPEL4WS unit testing: Test case generation using a concurrent path analysis approach. In: *International Symposium on Software Reliability Engineering (ISSRE)*, Los Alamitos, CA, USA: IEEE Computer Society, 2006, p. 75–84.

Zheng, Y.; Zhou, J.; Krause, P. An automatic test case generation framework for web services. *Journal of Software*, v. 2, n. 3, p. 64–77, 2007.

Zhu, H.; Hall, P. A. V.; May, J. H. R. Software unit test coverage and adequacy. *ACM Computing Surveys (CSUR)*, v. 29, n. 4, p. 366–427, 1997.

Appendix

A

Data Extraction Form

Table A.1 presents an example of the form filled with a study information. After collecting the necessary information for each selected study, the data were tabulated and analyzed.

Table A.1: Data extraction form filled with information of (Endo et al., 2008).

ID	25
Year	2008
Forum Name	Testing: Academic and Industrial Conference Practice and Research Techniques (TAIC PART)
Forum Type	Conference
Publication Subject	Software engineering and testing
Study Title	Web Services Composition Testing: a Strategy Based on Structural Testing of Parallel Programs
Author(s)	Endo, A. T.; Simão, A. S.; Souza, S. R. S.; Souza, P. S. L.
Application Context	Service Composition Testing – orchestration testing
Adopted Model	Control Flow Graphs
Empirical Study	Examples
Authorship	No other studies with shared authors.
References Connection	20
Original Abstract	Web Services have been used in the development of loosely coupled applications. Several Web Services are usually combined to create new services by a mechanism named Web Services Composition. In this paper, we present a strategy for Web Services Composition structural integra- tion testing. Structural testing coverage criteria for services written in BPEL are also described. The concept of required element groups is defined to improve the accuracy of crite- ria coverage. We present a case study for assessing the ap- plicability of proposed strategy. ValiBPEL-Web, a tool that supports the test strategy is also presented.
Reviewer Description	Endo et al. (2008) propose a test strategy to test BPEL based service composition. A test model for parallel programs named Parallel Control Flow Graph (PCFG) is adapted for representing the BPEL structures. The test model focuses mainly on the communication by means of the send and receive primitives. New types of definition and use are proposed and test criteria are defined for the associations. The concept of required element groups is proposed to increase the accuracy of the testing criteria. The architecture of the supporting tool called ValiBPEL-Web is also presented. The strategy is applied on three examples showing results about the applicability.