Regression Testing of Web Service: A Systematic Mapping Study

DONG QIU, BIXIN LI, and SHUNHUI JI, Southeast University
HARETON LEUNG, Hong Kong Polytechnic University

Web service is a widely used implementation technique under the paradigm of Service-Oriented Architecture (SOA). A service-based system is subjected to continuous evolution and regression testing is required to check whether new faults have been introduced. Based on the current scientific work of web service regression testing, this survey aims to identify gaps in current research and suggests some promising areas for further study. To this end, we performed a broad automatic search on publications in the selected electronic databases published from 2000 to 2013. Through our careful review and manual screening, a total of 30 papers have been selected as primary studies for answering our research questions. We presented a qualitative analysis of the findings, including stakeholders, challenges, standards, techniques, and validations employed in these primary studies. Our main results include the following: (1) Service integrator is the key stakeholder that largely impacts how regression testing is performed. (2) Challenges of cost and autonomy issues have been studied heavily. However, more emphasis should be put on the other challenges, such as test timing, dynamics, privacy, quota constraints, and concurrency issues. (3) Orchestration-based services have been largely studied, while little attention has been paid to either choreography-based services or semantic-based services. (4) An appreciable amount of web service regression testing techniques have been proposed, including 48 test case prioritization techniques, 10 test selection techniques, two test suite minimization techniques, and another collaborative technique. (5) Many regression test techniques have not been theoretically proven or experimentally analyzed, which limits their application in large-scale systems. We believe that our survey has identified gaps in current research work and reveals new insights for the future work.

Categories and Subject Descriptors: D.2.5 [**Software Engineering**]: Testing and Debugging

General Terms: Design, Experimentation, Reliability

Additional Key Words and Phrases: Regression testing, web service, test case prioritization, test case selection, test suite minimization

ACM Reference Format:
Dong Qiu, Bixin Li, Shunhui Ji, and Hareton Leung. 2014. Regression testing in web service: A systematic mapping study. ACM Comput. Surv. 47, 2, Article 21 (August 2014), 46 pages.
DOI: http://dx.doi.org/10.1145/2631685

1. INTRODUCTION

Web service is a widely used implementation technique under the paradigm of Service-Oriented Architecture (SOA). In current practice, service-oriented integration is a

This work is supported by the National Natural Science Foundation of China under No. 60973149, the College Industrialization Project of Jiangsu Province under Grant No. JHB2011-3, the Doctoral Fund of Ministry of Education of China under Grant No. 20100092110022, and the Scientific Research Foundation of the Graduate School of Southeast University No. YBJJ1313.
Authors' addresses: D. Qiu, B. Li, and S. Ji, School of Computer Science and Engineering, Computer Building 408, Southeast University, Nanjing, Jiangsu, P. R. China; email: {dongqiu, bx.li, shunhuiji}@seu.edu.cn. H. Leung, Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong; email: hareton.leung@polyu.edu.hk.

DOI: http://dx.doi.org/10.1145/2631685

mainstream application field of SOA, and the emergence of service composition technology makes the integration more convenient and efficient [Li et al. 2012].

A software system is subjected to continuous evolution due to modified system requirements. Regression testing is the de facto activity to address the testing problems caused by software evolution [Onoma et al. 1998]. A service-based system is no exception. After changes have been made to a system, regression testing can be performed to detect new faults that may have been introduced. Rerunning previously completed tests is a commonly applied strategy but it induces a high test cost. Alternatively, regression testing can be performed efficiently by systematically selecting the appropriate minimum set of tests needed to adequately cover a particular change or preferentially running tests with a higher rate of detecting possible faults. Since each invocation of a service may incur some charge, minimizing the number of payable invocations of services makes efficient regression testing more significant in web services.

Different from traditional software systems in which testers can easily identify the system changes, service testers might not even be aware of whether web services have been modified since they have no control over the evolution of *partner* services managed by the other service stakeholders [Canfora and Di Penta 2009]. Moreover, some inherent and specific characteristics (e.g., ultra-late binding mechanisms and nonobservability of the service implementation) make the web services regression testing more difficult [Bartolini et al. 2008; Mei et al. 2012]. Overcoming these challenges requires more adaptive and improved regression testing techniques. In the past 9 years (2005–2013), many research works have been presented. To better understand the key problems of this topic, we systematically studied existing methodologies/techniques and provided a comprehensive summarization.

Web service testing techniques have been surveyed and evaluated in several previous works [Canfora and Di Penta 2006; Palacios et al. 2011; Bozkurt et al. 2013]. However, most work surveyed multiple testing techniques in web service and no work concentrated on one special branch of testing techniques, especially regression testing techniques. Hence, it is indispensable to provide such a survey that comprehensively analyzes the current research state of this topic. To this end, we provide such a survey by answering the following questions: (1) *Who* participated in the regression testing activities? (2) *What* kinds of challenges were researchers facing and how many of them have been solved? (3) *What* kind of services and standards under services were addressed by regression testing techniques? (4) *Which* regression testing techniques were applied? (5) *How* did researchers validate their proposed regression testing techniques?

We performed a large-scale search on studies published from 2000 to 2013. Through a manual screening under careful reviews, a total of 30 papers have been selected as primary studies for answering our research questions. We scrupulously studied 60 regression testing techniques in total, from five perspectives of diverse service stakeholders. We validated these techniques and summarized seven possible research challenges in this topic. Based on our qualitative analysis, we have many solid findings. To the best of our knowledge, this is the first survey that concentrates on the topic of web service regression testing. We believe that our survey offers a guide on how to perform regression testing under the scenario of web service *w.r.t.* traditional regression testing. It comprehensively summarizes the research progress, identifies gaps, proposes new challenges in this field, and reveals new insights for future research directions. We hope this survey can also become a manual entry for other researchers to enter this emerging area.

The reminder of this article is organized as follows. Section 2 provides an overview of web service and regression testing techniques. Section 3 describes the review protocol used in our study. Section 4 presents the review results, and Section 5 discusses the

results, suggests directions of further study, and compares the related works. Finally, Section 6 concludes.

2. BACKGROUND

To supply the context for this mapping study, an overview of regression testing and web service is illustrated in the following subsections.

2.1. Regression Testing

Regression testing is triggered when changes have been made during software evolution. According to the *systems and software engineering vocabulary*[1] published in 2010, regression testing is *selective* retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements. The purpose of regression testing is to provide confidence that newly introduced modifications do not obstruct the behaviors of an existing, unchanged part of the software [Yoo and Harman 2012]. A number of approaches have been proposed for regression testing, and three major branches of the most widely used techniques have been identified: *test suite minimization (reduction)*, *test case selection,* and *test case prioritization*. Test suite minimization aims at obtaining a minimal subset of a test suite that preserves a specified adequacy criterion (e.g., coverage) [Rothermel et al. 2002]. Test case selection concerns reusing test cases from an existing test suite to test the modified part of the program [Rothermel and Harrold 1996]. Test case prioritization schedules test cases for execution in an order that attempts to increase their effectiveness at meeting some desirable properties, such as the rate of fault detection [Rothermel et al. 2001]. The three branches will be used as classification criteria of web service regression testing techniques in our review.

2.2. SOA and Web Service

SOA is a set of principles and methodologies that help in designing a software system to provide services to either end-user applications or to other services distributed in a network, via published and discoverable interfaces. Service is an SOA concept, and web service is a commonly used service technology at present, established upon a set of XML-based protocols including Simple Object Access Protocol (SOAP),[2] Web Service Description Language (WSDL),[3] and Universal Description, Discovery and Integration (UDDI).[4] Web services are usually classified into two groups: *basic* (or *atomic*) services and *composite* services. A basic service is an indivisible software component with exposed invocation connectors; a composite service aggregates smaller and fine-grained services and provides more powerful functionalities. Composite services may aggregate basic or other composite services.

Figure 1 gives the enhanced web service architecture, an extension of the traditional triangle model proposed in the group note *Web Service Architecture*[5] from the World Wide Web Consortium (W3C), showing the interactions between *five* different stakeholders. According to the granularity of web service, this architecture can be divided into two levels: *basic* level and *integration* level. At the basic level, *service developers* create *basic* services based on the desirable requirements or wrap legacy systems to expose functionalities as services. After services have been developed and deployed, they entrust *service providers* to manage the related affairs, including service registration

[1]ISO/IEC/IEEE 24765:2010, http://www.iso.org/iso/catalogue_detail.htm?csnumber=50518.
[2]http://www.w3.org/TR/soap12-part1/.
[3]http://www.w3.org/TR/wsdl20/.
[4]http://uddi.org/pubs/uddi_v3.htm.
[5]http://www.w3.org/TR/ws-arch/.

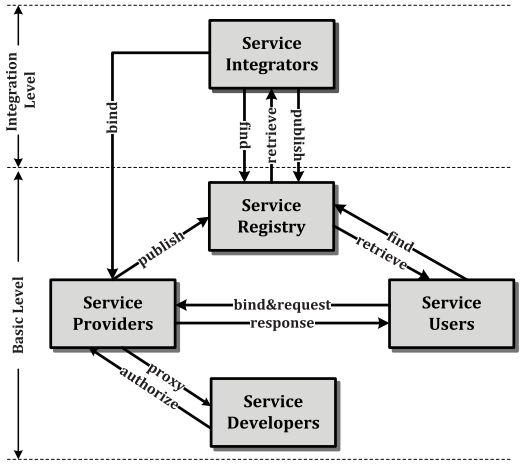

Fig. 1. Enhanced web service architecture.

and profit. Service providers then publish a description of service interfaces, generally specified in WSDL, in the *service registry*. After the services are published, *service users* can send a query request to find the potential services that complied with their requirement. The service registry matches users' requests with all information submitted and returns a list of service descriptions that meet the requirements to the service users. Service users select the most suitable service based on the evaluation results and bind with an offered endpoint to invoke the service and receive the corresponding result.

At the integration level, *service integrators* aggregate the functionality provided by the existing services to implement a value-added application. This is referred to as *service composition,* and the aggregated web service becomes a composite service. More specifically, service integrators define the structure of the composite service, generally called *process*, to satisfy the requirements of the integrated system. After the process is constructed, they query desired external services in the service registry and bind with the most reliable one to complement the functionalities that have been implemented by third parties. The procedure of querying and binding can be divided into two types: (1) *Static* binding, where both querying and binding are done manually at design time. The binding is tightly bound to one and only one service implementation before execution. Although it offers less flexibility on extending service functionalities, service integrators explicitly know which partner services they are using in this case. (2) *Dynamic* binding, where both querying and binding are done automatically at runtime. This approach leads to uncertainty of partner service selection, which naturally poses a great challenge for service integrators to guarantee the correctness of the whole composite service as the bound service may originate from any one of the candidates. After the process of composite service is defined, service integrators then publish its interface description in the service registry for other service users or other integrated service systems to invoke.

As Figure 1 depicts, different stakeholders of web service have different levels of access to artifacts to perform activities of regression testing. On the one hand, regarding the testing goals, service developers aim at ensuring the correctness of their developed services. Service providers try to ensure that the services and corresponding interface under their management meet the declarations stated by service developers. As third parties, service registries assess all registered services to provide fair assessments for service integrators and service users. For service integrators, they test to obtain

Table I. Research Questions

Ref.	Question
General Questions	
GQ1	Which of the different stakeholders are involved in the regression testing?
GQ2	What are challenges in web service regression testing? How many have been solved?
GQ3	What kinds of services and standards under services are addressed by regression testing techniques?
GQ4	Which regression testing techniques and methods are used?
GQ5	Which methods are used to validate the research?
Focused Questions	
FQ1	Which techniques are used to prioritize test cases in web service?
FQ2	Which techniques are used to perform test case selection in web service?
FQ3	Which techniques are used to reduce the size of test suite in web service?
Statistical Questions	
SQ1	How much activity about the research of web service regression testing has there been in recent years?
SQ2	Where have the researches been published?

confidence that any partner services to be bound to the composite service meet the requirements established at design time. Service users only require that service behavior satisfy their expectation. On the other hand, due to the restriction on accessing various resources, different stakeholders adopt different techniques to reach their goals. For example, as service integrators only have access to the interface information of candidate external services, only black-box testing techniques can be applied to these external services. Instead, service developers can use white-box testing techniques to test the functionality of their self-developed basic services. However, they cannot perform accurate nonfunctional testing since they cannot test the interactions between the invoked basic services and the composite services.

3. METHOD

The systematic mapping study (or mapping study) is one kind of *S*ystematic *L*iterature *R*eview (SLR), which is designed to provide a wide overview of a research area, establish if research evidence exists on a topic, and provide an indication of the quantity of the evidence [Kitchenham and Charters 2007]. We selected the mapping study as our survey approach because our goal is to summarize the current research state and identify more promising directions, which does not require an in-depth analysis and synthesis. We followed the methodology of SLR and referred to the guidelines proposed by Kitchenham and Charters [2007]. The following sections describe the process of our mapping study.

3.1. Research Questions

Specifying research questions is most important in any systematic review [Kitchenham and Charters 2007]. Our research questions are classified into three categories: General Question (GQ), Focused Question (FQ), and Statistical Question (SQ). Table I lists all research questions.

GQs concern universal problems of regression testing in SOA and web service. GQ1 refers to the question of *WHO* takes part in the testing process since testing resources are distributed among different stakeholders. This is the dominant factor that will directly influence the enactment of a regression testing strategy. GQ2 refers to the question of *WHY* regression testing techniques are needed in web service, especially

the new challenges that do not exist in the traditional software testing environment. GQ3 is related to the question of *WHAT* has been researched, not only the services (or service-based systems) under test (SUT), but also the underlying standards for implementing the SUT. GQ4 refers to the question of *HOW* to perform regression testing by specific techniques and methods. Finally, GQ5 considers trustworthiness and reliability of the proposed techniques.

FQs address the problems specific to regression testing techniques in web service. All questions in FQs are selected to help answer GQ4 from different perspectives. FQ1 concentrates on *HOW* test cases are prioritized to improve their capability of detecting faults. FQ2 focuses on *HOW* to perform the test case selection when changes of different types have been identified during the service evolution. FQ3 focuses on *HOW* to reduce the size of the test suite according to certain requirements.

From the perspective of the publication statistics, SQs offer another way to reflect on the quality of current research. SQ1 and SQ2 address the questions of when and where research papers were published. They are expected to provide not only the research trend but also the research maturity of this topic.

3.2. Search Strategy

The next step is to find the complete set of primary studies that are related to the research questions using an unbiased search strategy. This process involves a search strategy including construction of search keywords and definition of search scope.

3.2.1. Construction of Search Keywords. To obtain a better and more accurate search result, it is necessary to refine and optimize the group of keywords. Kitchenham and Charters suggested breaking down the search question into individual facets as search units where their synonyms, abbreviations, and alternative spellings are all included and then combined by Boolean operators [Kitchenham and Charters 2007]. A PIO (Population, Intervention, and Outcome) criterion is a suitable way to define such search units [Petticrew and Roberts 2005]. The *populations* involve terms that are related to the technologies and standards. Under the scenario of web service, we define the population keywords as follows:

Populations. (i) *Technology-related terms*. web service (ws),[6] Java web service, restful web service, service based, service centric, composite web service, composite service, business process, service-oriented architecture (SOA), service-oriented computing (SOC), orchestration, choreography. (ii) *Standard-related terms*. SOAP, WSDL, UDDI, BPEL [WS-BPEL, BPEL],[7] BPMN, OWL-S, WS-CDL, SLA, WS-agreement.

The *Interventions* address a specific issue in software methodology (or technology, tool, procedure). With respect to regression testing techniques covered in this review, we define the intervention keywords as follows:

Interventions. regression testing, regression test, test [test case, test suite] selection, test [test case, test suite] minimization {reduction},[8] test [test case, test suite] prioritization.

The *outcomes* are related to factors of importance to practitioners (e.g., improved reliability). With respect to regression testing of web service, it might refer to reduced testing cost, reduced time to perform testing, reduced number of test cases for maximum coverage of modification, and higher average percentage of faults detected by prioritized test cases. As it is not a required search unit to restrict the search results, we do not include the outcomes in the search terms.

[6]A (b) means that b is the abbreviation of a.
[7]A [b] means that b is an alternative spelling of a.
[8]A {b} means that b is a synonym of a.

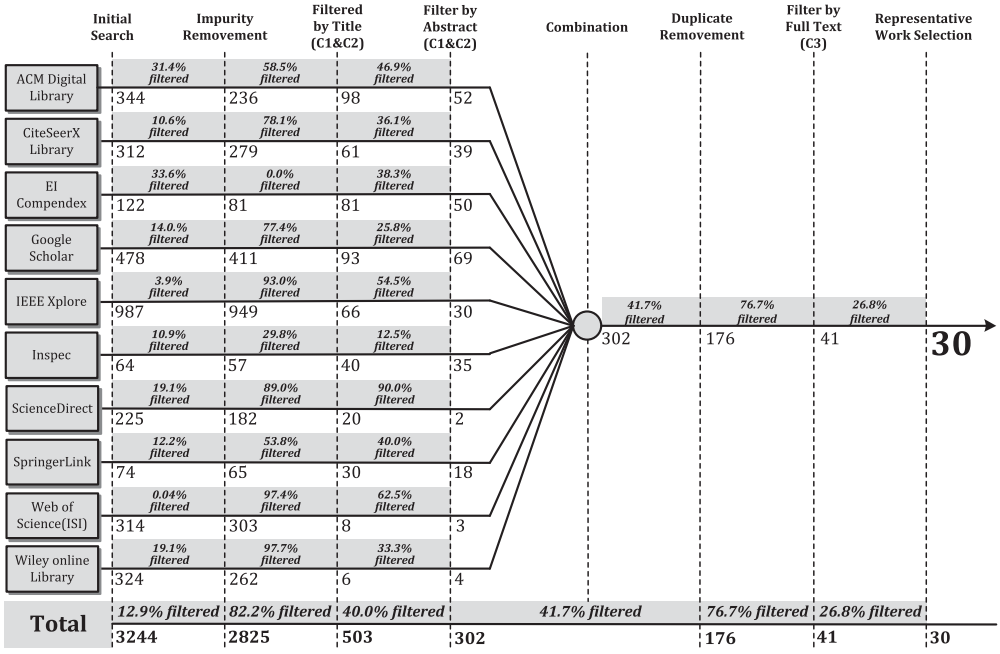

Fig. 2. The process of primary studies selection.

Hence, the final keyword set is: **Keywords** = **Populations** *AND* **Interventions**.

3.2.2. Source of Information. The source studies are obtained from selected electronic databases by searching with our constructed search keywords. To cover as many related studies as possible, we selected 11 electronic databases as our search scope, which are listed in Table XIV in Appendix A. These databases cover the most relevant journals and conferences (including workshops) within the area of computer science and engineering. Duplicated results produced from different databases are excluded by manual filtering in study selection.

To limit our search, we set the start year to 2000, in which the three key web service standards (SOAP, WSDL, and UDDI) were published by W3C. In addition, we set the end year to 2013, since our study was conducted in early 2014.

Finally, our initial search located 3,244 potentially relevant papers.

3.3. Study Selection

Next, we exclude the irrelevant studies from the initial search results and select the most representative studies as primary studies. The following exclusion criteria are applicable in this review, that is, exclude studies that:

$C1$: do not address the issue of testing.
$C2$: do not address the issue of web service.
$C3$: do not address the regression testing techniques.

Figure 2 shows the selection process, with the results after each filtering step.

(1) Remove impurities of the search result. Some impurities, for example, names of conferences (workshops) that are directly correlated to the search keywords, were included in search results due to characteristics of different electronic databases. We removed them manually to guarantee the accuracy of the results.

(2) Filter studies by C1 and C2. In this step, we applied C1 and C2 to the *title* and *abstract* of the studies to exclude studies that did not address either testing or web service technology.
(3) Group all remaining studies and remove duplicates. As some studies were found in more than one database, the duplicates were removed to guarantee the uniqueness of each study.
(4) Filter studies by C3. After step 2, some suspicious studies remained. We applied C3 to the *full text* of the studies to exclude studies that were not relevant to the topic of web service regression testing techniques.

Before applying the exclusion criteria, 3,244 papers were found from the initial search; 12.9% (419) of them were identified as impurities. After removing them, exclusion criteria C1 and C2 were applied to the remaining studies; 82.2% (2,322) were filtered out through title review and 40.0% (201) were filtered out through abstract review. The remaining studies from different electronic databases were then grouped together, and 41.7% (126) of them were identified as duplicates and removed. After this step, exclusion criterion C3 was applied to the full text of 176 studies and only 41 (26.8%) passed.

Looking into the short-listed 41 candidate studies, we found that some studies from the same author or research group are *technically similar*. Namely, one study was updated from a previous version where the methods or techniques illustrated are basically the same. Hence, for those studies along the same research line, we selected the most representative one as the primary study and removed the rest. Consequently, 11 studies were excluded. The process to select the most representative research work as the primary study is shown in Appendix B. Finally, a total of 30 papers were chosen as our primary studies.

3.4. Quality Assessment

It is critical to assess the quality of primary studies. The quality criteria are listed as follows, where most of them are adopted from Kitchenham and Charters [2007], Palacios et al. [2011], Afzal et al. [2009], and Engström et al. [2010]:

$Q1$: Is there a clear statement about the aim of the research?
$Q2$: Is there an adequate description of the research context?
$Q3$: Is there a review about the related work of problem?
$Q4$: Is there a description of the regression testing method or technique used in the research?
$Q5$: Has the approach been validated?
$Q6$: Is the conclusion related to the aim and purpose of research defined?
$Q7$: Is there a clear statement of findings?
$Q8$: Does the study recommend further research?

Table XV in Appendix C shows the result of applying the quality assessment criteria to each primary study where the ✓ indicates "yes" and the × indicates "no." All of our designed criteria are fulfilled by most of the primary studies except Liu et al. [2007], Dong [2008], and Tsai et al. [2009], which do not satisfy the related work review (Q3) and prospects of future work (Q8). We have decided not to eliminate them since missing Q3 and Q8 does not affect the study outcomes.

3.5. Data Extraction

Finally, we designed a data extraction form to collect information that addresses the research questions. Table XVI in Appendix D shows the detail of each extraction item

Table II. An Overview of Primary Studies

ID	Primary Study	Description
S1	Askarunisa et al. [2010]	test case prioritization based on coverage of service sequence
S2	Askarunisa et al. [2011]	test case prioritization based on fault rate and fault severity
S3	Athira and Samuel [2010]	test case prioritization based on coverage of service activity
S4	Bai and Kenett [2009]	test case ranking based on the risks of target service features
S5	Bozkurt [2013]	cost-aware test suite minimization based on multiobjective optimization
S6	Chen et al. [2010]	test case prioritization based on the weighted service activity
S7	Di Penta et al. [2007]	collaborative regression testing technique
S8	Dong [2008]	test case reduction based on pairwise combination of input parameter
S9	Hou et al. [2008]	scheduled test case prioritization for quota-constrained service-centric system
S10	Li et al. [2008]	gray-box test case selection based on BFG model for business process
S11	Li et al. [2012]	test case selection based on XBFG model for composite web service
S12	Liu et al. [2007]	test case selection based on BFG model for business process
S13	Khan and Heckel [2011]	test case selection based on dependency analysis of visual contracts
S14	Mei et al. [2009]	test case prioritization based on multilevel coverage models
S15	Mei et al. [2009]	test case prioritization based on the coverage of WSDL tags
S16	Mei et al. [2011]	test case prioritization based on the occurrence of WSDL tags
S17	Mei et al. [2012]	preemptive scheduled test case prioritization for adaptive services
S18	Mei et al. [2013a]	test pair prioritization based on the structural similarity of XML artifacts
S19	Mei et al. [2013b]	similarity-based test case prioritization techniques based on pairwise selection
S20	Nguyen et al. [2011a]	test case prioritization based on change sensitivity of test case
S21	Nguyen et al. [2011b]	IR-based test case prioritization based on covered identifier documents
S22	Ruth et al. [2006]	test case selection based on JIG model for Java-based web service
S23	Ruth et al. [2007][a]	test case selection based on global CFG and call graph for composite web service
S24	Ruth [2008]	test case selection on handling multiple concurrent modifications
S25	Ruth [2011]	test case selection on the premise of protecting the privacy of shared services
S26	Ruth and Rayford [2011]	test case selection on the premise of no sharing of service privacy
S27	Tarhini et al. [2006]	test case selection based on TPG and Timed LTS model
S28	Tsai et al. [2005]	test case ranking based on the probability of detecting faults
S29	Tsai et al. [2009]	test case ranking based on potency and coverage relationship model
S30	Zhai et al. [2014]	point-of-interest-aware test case prioritization for location-based services

[a]Ruth and Tu [2008] is also a part of primary study S23.

for the primary studies. This form enabled us to extract full details from the primary studies and understand how the studies addressed our research questions.

4. RESULTS

In this section, we present results from 30 assessed primary studies related to our research topic. An overview of all primary studies is shown in Table II where id, citation, and the basic description of each study are included. We then answer our research questions in the following subsections through elaborative information synthesis.

4.1. Stakeholders

In web service testing, many service stakeholders are involved in the testing process to different degrees [Tsai et al. 2004]. However, since stakeholders from diverse perspectives have different accessibility of service resources, their testing strategies and emphasis may be different. There is no exception on web service regression testing. It is an essential task to figure out the testing duties of all participants when services evolve. We refine the definition of stakeholders presented in Canfora and Di Penta [2006] under the scenario of regression testing according to the enhanced web service architecture shown in Figure 1.

—*Service developers*. They focus on the quality of the basic services they have developed. Since they can access the implementation of services, all regression testing techniques in traditional software testing can be adopted, especially the white-box testing techniques [Bozkurt et al. 2013]. In addition, they can also use the black-box techniques to test the interface of basic services. In general, all changes made to the basic services are driven by service developers, who can perform regression testing immediately after any modification on basic services.
—*Service providers*. They focus on the conformance between the service capabilities and requirements. Different from service developers, they can only access the interface of services. When the implementation of services evolves, they may not be aware of the changes if no notification is received. This requires continuous monitoring to detect the evolution of services and immediate regression testing to guarantee the conformance. Alternatively, a notification mechanism between service providers and service developers can be established to maintain such an *invisible* evolution.
—*Service registries*. They focus on evaluating the actual quality of services registered in their servers and providing a fair third-party assessment to the service integrators or the service users. They manage and update the quality parameters of services at regular intervals to keep the assessment status up-to-date. Since they can only access the interface of services, they face the same challenges as those of service providers.
—*Service integrators*. They are concerned with checking whether new faults have been introduced on both process and partner services when changes are made to the composite services. Since they have access to the implementation of the process, all white-box regression testing techniques can be adopted to reveal the faults in the process. However, they do not have any control over the changes in partner services. This requires service integrators to monitor errors in system behavior or changes in the system performance to detect the partner services' evolution at runtime if they cannot receive any notification from the service providers [Bozkurt et al. 2013]. Moreover, composite services may self-evolve automatically that adaptively rebind new partner services to meet the given nonfunctional requirements (e.g., response time) during the execution. In this case, service integrators also need to monitor the runtime behavior after service replacement to ensure the correctness under the new binding.
—*Service users*. They focus on whether services offer the expected functions. In most cases, they do not care whether services are modified if the changes do not affect their usage experience. In addition, they would be more likely to obtain the evaluation results from service registries instead of doing testing themselves.

Figure 3 depicts the scope of control of all stakeholders through an example of composite service *CS1*. Suppose basic service *BS1* is created by the service developer and managed by the service provider. *I1* is the implementation of *BS1*. The service integrator constructs composite service *CS1* where *A1*, *A2*, . . . , *A6* are activities that

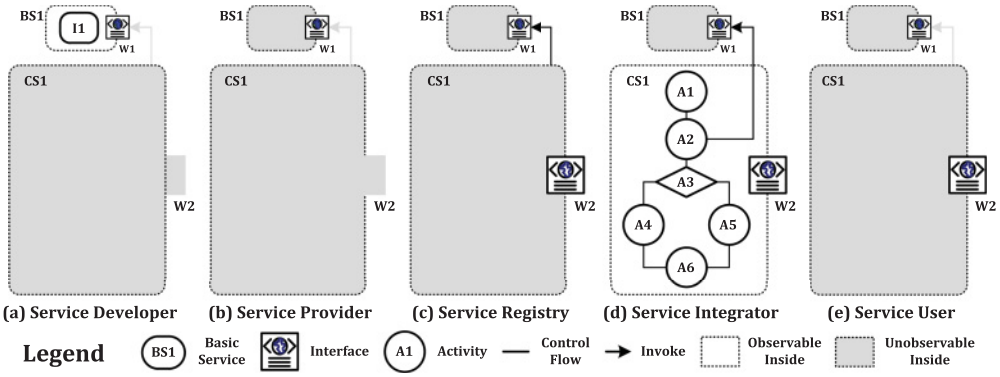

Fig. 3. Service observability for different stakeholders.

constitute the process of *CS1*. *BS1* is invoked by *CS1* as its partner service. *W1* and *W2* are interfaces of service *BS1* and *CS1*, respectively. Looking at each subgraph in Figure 3, resources covered with gray are not under the control of the corresponding stakeholders. Consider (d) as an example: the service integrator cannot access *I1* of *BS1*. He can only access its interface *W1*. The constraints of accessing unshared test objects directly affect the enactment of the testing strategy.

For research question GQ1 (Which of the different stakeholders are involved in the regression testing process?), statistical results from the primary studies are shown in Table III. Service integrator is the most frequently cited stakeholder emerging in about three quarters (76.7%, 23/30) of the primary studies, whereas service provider is cited in more than a quarter of the studies (26.7%, 8/30). Among these studies, the unavailability or partial unavailability of the service implementations are the key factors that make regression testing from these perspectives more challenging. Only one study suggests that the involvement of service registry is also beneficial to the whole testing process. No study considers the issue from the perspective of the service developer and service user. It is noteworthy that study S7 invited diverse stakeholders, including service provider, service integrator, and service registry, to prepare a regression testing plan in a collaborative manner, for the purpose of resource sharing.

In summary, most research emphases have been put on how to perform regression testing from the perspective of *service integrators*. The uncontrollability over partner services' evolution and service rebinding makes the process of regression testing highly uncontrollable and uncertain. This presents a greater challenge to the regression testing from this perspective compared to other perspectives. Very few studies put their emphasis on the service developer. The possible reason is that challenges from this perspective are quite similar to the ones in traditional regression testing. Although the perspective of service registry is not attractive currently, it may induce some promising research directions, including (1) how to efficiently manage services and the corresponding test suite in a unified manner to improve the sharing of the testing resources, and (2) how to convene more stakeholders in the collaborative activities to reduce the costs and difficulties of the regression testing.

4.2. Challenges of Regression Testing in Web Service

Web service regression testing is generally accepted to be more challenging compared with regression testing of traditional systems [Canfora and Di Penta 2006]. Identifying the specific challenges can greatly help to select the most suitable testing techniques

Table III. Distribution of Primary Studies by Stakeholders

Stakeholders		Primary Studies																													
		S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	**Total**
Developer																																**0**
Provider		✓						✓		✓				✓							✓	✓							✓	✓		**8**
Registry								✓																					✓			**2**
Integrator			✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓			✓	**23**
User																																**0**

or design new testing methods. Regarding the general questions GQ2 (What are challenges in web service regression testing? How many have been solved?), we extract all potential challenges from each primary study by reviewing the sections of *abstract*, *introduction,* and *background/preliminary* (if it exists). Then, reviewing the main content and *conclusion* of the primary study, we try to ascertain how well each identified challenge has been solved. The challenges are presented from the following seven aspects, corresponding to different issues specific to web service regression testing:

—*Cost (Ch1).* Efficiently and effectively performing regression testing in web services is always a challenge, which is still an open problem in traditional software regression testing. How to minimize the test suite, select test cases with high fault detection capability, and preferentially run test cases with early fault detection greatly affects the test cost. In addition, under the scenario of SOA, many services, especially those developed by third-party organizations for profit, are not always free of charge for service integrators and users [Di Penta et al. 2007; Khan and Heckel 2011]. If fees of service are paid on a per-use basis, frequent invocations of chargeable service may entail a high test cost, which makes the constraints on the number of test cases executed much stricter than those of traditional software [Nguyen et al. 2011b]. Thus, how to minimize the number of payable invocations to reduce the test cost should be addressed.

—*Autonomy (Ch2)*. Complex service-based systems are usually composed of many partner services, which are mostly developed by third parties and reside in remote locations [Bozkurt et al. 2013]. If permission is not authorized, no other stakeholders except service developers can access their implementation. Hence, the unavailability of partner services' implementation makes code-based approaches, especially white-box testing techniques, inapplicable in web service regression testing [Mei et al. 2011]. In addition, since testers cannot easily obtain the details of partner services, they can only request them though the Internet, which also leads to the challenge of *Ch1*.

—*Test timing (Ch3)*. Traditionally, regression testing is performed immediately after changes have been made to software systems. However, the inherent distributed characteristics of web service, autonomy, and loose coupling greatly affect the choice of test time. That is, all participating services in the main system are usually developed and maintained independently by different organizations and composed on the syntactical match of the interfaces. As long as a service does not change its interface, the consumer will not be aware of any implementation change made to the service [Li et al. 2008]. Such uncontrollability over service evolution leads to a critical timing problem for performing regression testing if service stakeholders cannot receive any prior notification about participating services' evolution [Mei et al. 2012].

—*Dynamics (Ch4)*. Web service can dynamically change its internal processing logic or bind to new external services during the course of an execution. Such ultra-binding and adaptive service replacement are called *late change* [Mei et al. 2012]. A new challenge arises when a late change has been detected during the process of regression testing: what is the best strategy to continue testing?

—*Concurrency (Ch5)*. This issue is always mentioned in distributed systems, as well in service-based systems. Different services may be modified concurrently and independently because of their autonomy [Ruth and Tu 2007a]. However, the lack of complete understanding of the global modifications may cause the test inconsistency, which has a strong impact on the correctness of the testing results. Consider Figure 3(d) as an example; the activity *A2* in the composite service *CS1* invokes the partner service *BS1*. Suppose some modification happens in *A2* and it subsequently activates a regression testing process to check whether change in *A2* will affect other parts of

CS1. When test cases are being executed, another modification is made to *BS1*. This leads to inconsistency in testing where part of test cases are running on the old *BS1* and the rest are on the new *BS1*. Other possible scenarios were discussed in [Ruth and Tu 2007a; Ruth 2008]. This issue also makes *fault location* in web services more difficult [Mei et al. 2008; Bozkurt et al. 2013].

—*Quota constraints (Ch6)*. Service providers often impose some constraints on public services (e.g., Google SOAP Search API) or even payable services (e.g., Amazon Historical Pricing Web Service) to avoid responding to superfluous web service requests, such as constraints of network flux, storage usage, and request quotas [Hou et al. 2008]. The request quota defines the upper limit of the number of requests that a user is permitted to send to a web service during a certain time period. However, the regression testing of large-scale service systems is usually intensive, which may easily consume the quota of invoked services. Then, fault exposure and the subsequent debugging will be delayed and testing cost may increase. Hence, how to efficiently expose faults given the quota constraint becomes a challenge.

—*Privacy (Ch7)*. Because of the challenge *Ch1* that part of participating services' implementations are not open to testers, regression testing on composite services is often inadequate and incomplete. To tackle this problem, collaborative regression testing [Di Penta et al. 2007] is introduced. It invited diverse stakeholders to take part in the testing activities and shared useful information of the participating services, such as structures, test cases, and coverage information of corresponding test cases [Ruth 2011]. However, such a collaborative model is inapplicable when sensitive information of services may be leaked across multiple services to some participants. Therefore, finding the balance between securing sensitive information and improving test integrity is really challenging.

Table IV shows, for each primary study, the specific challenges the studies attempted to conquer. It indicates that most studies (80%, 24/30) concentrate on *Ch1*. Among them, most studies proposed approaches to minimize the size of the test suite or improve the capability of fault detection. Only one study S5 considered the charge of service invocations as one of the objectives to minimize the test suite. Eleven studies focus on *Ch2* to improve the testing completeness based on partial knowledge of service implementation. Very few studies (around one to two) work on each of the rest of the five challenges, which are all specific to web service regression testing. The result reveals there are still many challenges not fully addressed.

4.3. Services Under Test and Related Standards

Understanding the scope of regression testing techniques can help service stakeholders select suitable techniques and apply them to specific types of services and standards underlying services. To answer GQ3 (What kinds of services and standards underlying services are addressed by regression testing techniques?), we studied each primary study carefully and summarized the concrete service types and standards targeted by each study. In addition, we classified the standards underlying the SUT into two groups: (1) the service interfaces and (2) the internal and external behavior of composite service. The results are shown in Table V.

Most of the studies (93.3%, 28/30) focused on composite services, while one-quarter of the studies (26.7%, 8/30) focused on basic services. Besides, six studies considered both kinds of services. In most of these six studies, only interface descriptions are prerequisite and no information specific to service type (e.g., structure of composite services) is required [Tsai et al. 2009; Zhai et al. 2014].

Regarding the service description, more than half of the primary studies (63.3%, 19/30) required WSDL as the default interface standard. Mei et al. used *WSDL tag* (i.e.,

Table IV. Challenges That Have Been Addressed by Primary Studies

		Primary Studies																														
		S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	**Total**
Challenges	Ch1	✓	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	**24**
	Ch2	✓									✓	✓		✓	✓	✓	✓		✓	✓	✓	✓										**11**
	Ch3							✓																								**1**
	Ch4																	✓														**1**
	Ch5																								✓							**1**
	Ch6									✓																						**1**
	Ch7																									✓	✓					**2**

Table V. Distribution of Primary Studies by SUT and Underlying Standards

			Primary Studies																														
			S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	**Total**
SUT	Basic Service						✓		✓		✓				✓							✓								✓	✓	✓	**8**
	Composite Service		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**28**
Standards	Service Description	WSDL	✓	✓	✓	✓			✓			✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓					✓		✓		**19**
		OWL-S																															**0**
		Other													✓										✓	✓	✓	✓		✓			**6**
	Composite Behavior	BPEL						✓		✓		✓	✓	✓		✓	✓	✓	✓	✓	✓											✓	**12**
		WS-CDL																															**0**
		OWL-S		✓		✓																									✓		**3**
		Other	✓		✓																		✓	✓	✓	✓	✓	✓	✓				**9**

XML elements) to enact the prioritization strategies Mei et al. [2009, 2011]. Besides, Li et al. [2012] calculated the evolution of the service interfaces by comparing the old and new versions of WSDL documents. Six studies used other informal standards, for example, visual contract [Khan and Heckel 2011] and CFG [Ruth et al. 2007; Ruth 2008, 2011; Ruth and Rayford 2011]. Five studies did not mention the standards used.

Regarding the behavior of the composite services, BPEL[9] is the most widely adopted language in both the academic and industrial community, which has become the de facto standard for web service orchestration. Nearly half of the studies (40%, 12/30) used BPEL to specify the behavior of composite services. Li et al. [2008, 2012] considered the control flow relation for BPEL-based service while Liu et al. [2007] addressed the issue of concurrency. Three studies focused on the composite services that used OWL-S to specify their semantic behaviors. Nine studies used other informal standards or user-defined structure to define composite behavior. Ruth et al. [2007] generated the *global CFG* for composite services from the CFGs of all partner services based on the composite structure. Four studies did not explicitly specify any standard because their approaches are independent of concrete standards or protocols.

In summary, our findings include the following: (1) regression testing on composite service is a research hotspot as it involves more challenges to overcome; (2) WSDL is popular in specifying the interface of web services and many regression testing techniques rely on it; and (3) orchestration-based services, especially those specified in BPEL, have been extensively studied. However, no attention has been paid to regression testing on either semantic-based services (e.g., OWL-S) or choreography-based services (e.g., WS-CDL).

4.4. Regression Testing Techniques

As discussed in Section 2.1, test suite minimization, test case selection, and test case prioritization are adopted as the classification of web service regression testing techniques in this review. Table VI gives the techniques used by each primary study. More than half of studies (56.7%, 17/30) used test case prioritization techniques to optimize the execution order of test cases to achieve a higher fault detection rate. Ten studies applied test case selection to choose as few test cases as possible to cover all areas affected by modification. The test suite minimization techniques were adopted in only two studies. In particular, one study [Di Penta et al. 2007] mentioned a framework of managing multiple stakeholders in the regression testing process. The research question GQ4 (Which regression testing techniques and methods are used?) is answered in the following subsections.

4.4.1. Test Case Prioritization. Since the goal of test case prioritization is to find a permutation order of test cases such that test cases with a higher fault detection capability will be executed earlier, the design and selection of prioritization techniques play a crucial role. Regarding the focus question FQ1 (Which techniques are used to prioritize test cases?), we have extracted 48 prioritization techniques from the primary studies. Table VII provides an overview of these techniques, with their *mnemonic* name, *description,* and *source* listed. We classify them into five groups: *benchmark*, *coverage based*, *fault-exposing-potential (FEP) based*, *information retrieval (IR) based,* and *other* prioritization techniques.

Benchmark prioritization techniques. Two classic prioritization techniques are (1) P1 randomly orders the test cases and (2) P2 optimally orders the test case with the maximum rate of fault detection. Although P2 is an ideal but not viable approach

[9]http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.

Table VI. Distribution of Primary Studies by Regression Testing Techniques

Techniques	Primary Studies																														
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	**Total**
Minimization					✓			✓																							**2**
Selection										✓	✓	✓	✓									✓	✓	✓	✓	✓	✓				**10**
Prioritization	✓	✓	✓	✓		✓			✓					✓	✓	✓	✓	✓	✓	✓	✓							✓	✓	✓	**17**
Others							✓																								**1**

Table VII. Test Case Prioritization Techniques Collected from Primary Studies

Group	Ref.	Mnemonic	Description	Source
Bench-mark	P1	random	random ordering	
	P2	optimal	order to optimize rate of fault detection	
Cover-age based	P3	total-ac	coverage of service activities	
	P4	addtl-ac	coverage of service activities not yet covered	
	P5	total-tr	coverage of service transitions	
	P6	addtl-tr	coverage of service transitions not yet covered	
	P7	total-wtd-ac	coverage of weighted process activities	S6
	P8	addtl-wtd-ac	coverage of weighted process activities not yet covered	S6
	P9	desc-seq	coverage of invoked services in descending order	S1
	P10	asc-seq	coverage of invoked services in ascending order	S1
	P11	desc-wt	coverage of WSDL tags in descending order	S15
	P12	asc-wt	coverage of WSDL tags in ascending order	S15
	P13	desc-wtc	coverage of WSDL tag counts in descending order	S16
	P14	asc-wtc	coverage of WSDL tag counts in ascending order	S16
	P15	total-cm1	coverage of workflow branches (*WB*s)	S14
	P16	addtl-cm1	coverage of *WB*s not yet covered	S14
	P17	total-cm2-s	coverage of *WB*s and XRG branches (*XRGB*s)	S14
	P18	addtl-cm2-s	coverage of *WB*s and *XRGB*s not yet covered	S14
	P19	total-cm2-r	*P13* with additional coverage of *WB*s for TC with same priority	S14
	P20	addtl-cm2-r	*P14* with additional coverage of uncovered *WB*s for TC with same priority	S14
	P21	total-cm3-s	coverage of *WB*s, *WRGB*s, and WSDL elements (*WE*s)	S14
	P22	addtl-cm3-s	coverage of *WB*s, *WRGB*s, and *WE*s not yet covered	S14
	P23	total-cm3-r	*P17* with additional coverage of *WE*s for TC with same priority	S14
	P24	addtl-cm3-r	*P18* with additional coverage of uncovered *WE*s for TC with same priority	S14
	P25	xps-total-sim	repeatedly select test case pair with maximum similarity	S18
	P26	xps-total-dsim	repeatedly select test case pair with minimum similarity	S18
	P27	xps-iter-dsim	iteratively select test pair from group with same similarity in ascending order	S18
	P28	xps-iter-sim	iteratively select test pair from group with same similarity in descending order	S18
	P29	xps-r-iter-dsim	*P28* that requires test pair selected contains one previous selected test case	S19
	P30	xps-r-iter-sim	*P29* that requires test pair selected contains one previous selected test case	S19
	P31	total-qc	coverage of testing quota-constrained requirements	S9
	P32	addtl-qc	coverage of testing quota-constrained requirements not yet covered	S9
	P33	fix	runtime reprioritize the not-yet-covered TCs when old item(s) are missed covered	S17
	P34	reschd	runtime reprioritize the not-yet-covered TCs when new item(s) are covered	S17
	P35	fix-reschd	hybrid of *P33* and *P34*	S17

Continued

Table VII. Continued

Group	Ref.	Mnemonic	Description	Source
	P36	potency	probability of detecting a fault	S28
	P37	potency-crm	probability of detecting a fault based on CRM	S29
FEP	P38	risk	risks detected on service with failure probability and importance	S4
based	P39	c-sensitivity	rate of detected changes over all changes in service response	S20
	P40	fault-rate	rate of the total number of faults detected to the execution time	S2
	P41	severity	combination of f-rate in *P36* and fault impact	S2
IR	P42	cm-i	covered identifier documents (class, method signature, and terms in signatures)	S21
based	P43	cmr-i	covered identifier documents in *P39* and requests content by services	S21
	P44	var	variance of a location sequence	S28
	P45	entropy	complexity of geodesic curve connected by the location sequence	S28
Others	P46	cdist	distance from the centroid of a location sequence to the centroid of a set of POIs	S30
	P47	pdist	mean distance of the set of POIs associated with the expected output to the polyline	S30
	P48	pcov	coverage of the polyline (calculated by *P44*) that is no greater than a value	S30

to rank test cases, it provides an upper bound on the effectiveness of other heuristics [Elbaum et al. 2002]. Both of them are the most commonly used techniques as basic comparative objects in almost all of the primary studies.

Coverage-based prioritization techniques. The basic idea originates from traditional software testing that selected test cases should execute or cover the whole program under test [Adrion et al. 1982]. Determining a quantitative measure of coverage is an indirect measure of system quality. Achieving higher coverage is generally considered essential to produce a higher-quality system. There is no exception in web service, where coverage-based techniques rely on the coverage information of service-related elements. Most of the prioritization techniques (68.8%, 33/48) in this survey were coverage based, among which P3–P30 presented prioritization techniques that focus on how to cover significant service-related elements to improve the ability of fault detection, while P31–P35 were frameworks for better applying existing coverage-based prioritization techniques (P3–P30) to the service systems with specific requirements. We illustrate the two groups separately as follows.

Normally, each test case is assigned a priority, calculated based on some predefined *criterion* and *coverage strategy*. The test case with the highest priority ranked by the *sort rule* then will be selected first to execute. In particular, test cases with the same priority need to be reordered by the *additional rule*. The first five columns in Table VIII list these details.

Regarding the coverage criterion, Service Activities (SAs) and Service Transitions (STs), which originated from statements and branches in traditional software artifacts, are the most commonly used criteria for comparison by most prioritization techniques. On this basis, Chen et al. [2010] assigned each SA a *contribution* in order to differentiate their significance. The test case covering more Weighted SA (Wtd. SA) had a higher priority than others. P9 and P10 used coverage of invoked Services (Ss) to prioritize test cases. P11–P14 introduced a new metric, called *WSDL tag* (WT), which is any XML element defined in XML schema in WSDL documents [Mei et al. 2009, 2011]. The coverage of WSDL tags can reveal the usage of the internal message

Table VIII. Comparison of Different Coverage-based Prioritization Techniques

Ref.	Criterion	Cov. Stra.	Sort Rule	Addtl. Rule(s)	White/ Black	General/ Specific	Granularity
P3	SA	TG	desc	random	white	general	P
P4	SA	AG	desc	random	white	general	P
P5	ST	TG	desc	random	white	general	P
P6	ST	AG	desc	random	white	general	P
P7	wtd. SA	TG	desc	random	white	general	P
P8	wtd. SA	AG	desc	random	white	general	P
P9	S	TG	desc	random	black	general	M
P10	S	TG	asc	random	black	general	M
P11	WT	TG	desc	random	black	general	I
P12	WT	TG	asc	random	black	general	I
P13	WTO	TG	desc	random	black	general	I
P14	WTO	TG	asc	random	black	general	I
P15	WB	TG	desc	random	white	general	P
P16	WB	AG	desc	random	white	general	P
P17	WB/XRGB	TG	desc	random	white	general	P/M
P18	WB/XRGB	AG	desc	random	white	general	P/M
P19	WB	TG	desc	XRGB(TG)	white	general	P/M
P20	WB	AG	desc	XRGB(AG)	white	general	P/M
P21	WB/XRGB/WE	TG	desc	random	white	general	P/M/I
P22	WB/XRGB/WE	AG	desc	random	white	general	P/M/I
P23	WB	TG	desc	XRGB/WE(TG)	white	general	P/M/I
P24	WB	AG	desc	XRGB/WE(AG)	white	general	P/M/I
P25	Similarity	PS	desc	random	white	general	P/M/I
P26	Similarity	PS	asc	random	white	general	P/M/I
P27	Similarity	PS	desc	none	white	general	P/M/I
P28	Similarity	PS	asc	none	white	general	P/M/I
P29	Similarity	PS	desc	none	white	general	P/M/I
P30	Similarity	PS	asc	none	white	general	P/M/I
P31	*optional*	TG	-	-	-	general	-
P32	*optional*	AG	-	-	-	general	-
P33	*optional*	-	-	-	-	general	-
P34	*optional*	-	-	-	-	general	-
P35	*optional*	-	-	-	-	general	-

among SA. Specifically, P11 and P12 counted multiple occurrences of the same WT as once for the computation of coverage of test cases. Somewhat differently, P13 and P14 counted the number of WT occurrences (WTOs) to differentiate tie cases with the same WT coverage. Besides, Mei et al. [2009] also proposed metrics of workflow branches (WBs), XPath Rewriting Graph branches (XRGBs), and WSDL elements (WEs) to calculate the coverage of test cases. Some techniques, such as P17, P18, P21, and P22, applied a combination of ST, XRGB, and WE as the coverage criteria. In addition, Mei et al. [2013a, 2013b] considered the structural relationships among the coverage of WB, WE, and XML message and further calculated the similarity based on the combined coverage information. The similarity provides the alternative information to distinguish test cases that could not be achieved using previous coverage-based techniques.

Regarding the coverage strategies, *Total Greedy (TG)* and *Additional Greedy (AG)* search algorithms were adopted in two-thirds (66.7%, 22/33) of the coverage-based techniques. Total-based techniques prioritize test cases based on information available from the initial state, with no consideration of the effect of prioritized test cases, whereas additional-based techniques pick a test case with the maximal coverage of items not yet covered by previously prioritized test cases and adjust the order of remaining test cases

by eliminating the effect caused by the selected test case. In addition, the *Pairwise Selection (PS)* strategy was adopted in six prioritization techniques based on the test case similarity. No other search algorithms, such as hill climbing and genetic algorithms [Li et al. 2007], have been adopted.

Regarding the sort rules, *descending* and *ascending* order are used to rank the priority of test cases. When multiple test cases have the same value, most techniques selected one of them randomly. There are exceptions for four techniques where additional rules were provided. Consider P19 as an example: when test cases produced the same total coverage of WB, total coverage of XRGB was used to break the tie.

Different from P3–P30, P31–P35 were independent of any concrete coverage-based techniques. They concentrated on how to apply current prioritization techniques to the service systems with specific requirements. P31–P32 were used for service systems that had requested quota constraints (i.e., a limit on the number of requests that a user can send to a web service during a certain time range) [Hou et al. 2008]. They divided the testing time into time slots to align different time ranges for different request quotas and prioritized test cases for each time slot by using existing prioritization techniques (P5 and P6 were adopted as samples in their case studies). P33–P35 were used for the adaptive composition services in which part of external services may evolve or be replaced by new services dynamically without prior notification [Mei et al. 2012]. They detected *late changes* during the execution of a regression test suite, preempted the execution, selected prioritized test cases from a regression test suite as fixes, ran the fixes, and then resumed the suspended execution of the regression test suite.

Based on these illustrations, we further compare these prioritization techniques from three dimensions. The first dimension is whether the prioritization technique is black box or white box. From the sixth column of Table VIII, six strategies were based on *black-box* techniques and the other 22 were based on *white-box* techniques.

The second dimension is whether the prioritization technique is *general* or *specific* [Rothermel et al. 2001]. The key difference between them is the usage of a specific version's modification information on prioritizing test cases. In the former case, given program P and test suite T, test cases in T are prioritized with the intent of finding an ordering of test cases that will be useful over a series of subsequent modified versions of P. In the latter case, given program P and test suite T, test cases in T are prioritized with the intent of finding an ordering that will be useful on a specific version P' of P, which means that the prioritization strategy is not general to evolution across all versions. From the seventh column of Table VIII, all of the prioritization techniques are general. They are effective to a succession of subsequent releases no matter how the services under test change.

The last dimension is the granularity test cases cover, considered here in terms of *process* (P) level, *message (M)* level, and *interface (I)* level. The process level is the highest level that describes the business logic of the composite services. The message level defines the interactions between all participating services. The lowest level, the interface level, provides information on how to access these services. The eighth column of Table VIII shows the granularity of each coverage-based technique. Granularity affects the relative cost of techniques in terms of computation and storage [Elbaum et al. 2002]. For example, as the coverage criterion of P21 (ST/XRGB/WE) is more fine-grained than that of P17 (ST/XRGB), the time required for prioritizing test cases by P21 is longer than that of P17.

As Table VIII shows, the criteria for P31–P35 are *optional* since they can adopt any suitable technique from P3–P24. The symbol "-" represents that its value depends on the selected concrete prioritization technique.

FEP-based prioritization techniques. The ability of a test case in detecting faults depends on not only the execution of a faulty component but also the probability that a fault in that component will cause a failure for that test case [Elbaum et al. 2002]. In practice, the calculation of such probability must be an approximation. Many approximation methods were proposed to rank the test cases. One approach estimates the probability of detecting fault for each test case. Tsai et al. [2005] used *potency* (P36), the probability of a test case to detect a fault based on the predicted test oracles, to rank the test cases. They improved the computation of potency (P37) by introducing the *coverage relationship model (CRM)* [Tsai et al. 2009], which provided the coverage probability between test cases and their execution states. Bai and Kenett [2009] prioritized test cases by estimating the *risk* (P38) of their target service features. The risk of a service feature is defined by two factors: the probability to fail and the consequence of the failure, where the former can be obtained from the structure-based analysis and the latter can be obtained from the dependence analysis and the usage analysis. In addition, *change sensitivity* (P39) was proposed to measure the importance of test cases based on the assumption that sensitive test cases that potentially execute more service changes have a higher ability to reach the faults caused by changes [Nguyen et al. 2011a]. They adopted several mutation operators to mutate services and calculate the change sensitivity based on the proportion of mutants that are killed (if mutated response of service differs from the original response).

Another approach is to simulate the capability of detecting faults by seeding faults. Askarunisa et al. [2011] prioritized test cases by their *fault rate* (P40), that is, detected seeded faults *w.r.t.* the execution time. In addition, *fault severity* (P41) was also proposed based on the combination of fault rate and fault impact (importance of a fault) [Kavitha and Sureshkumar 2010]. P41 seems to offer a stronger ability of fault detection than P40 and other coverage-based techniques, such as P3, P5, P9, and P11.

IR-based prioritization techniques. The service execution trace and service evolution information may be related. The hypothesis on IR-based techniques is that if the execution history of one test case covers more *identifiers* emerged simultaneously in the service change descriptions, the test case has a higher probability to cover the potential faults caused by this change. Based on this hypothesis, the degree of matching between service change query and service execution history can be used to measure the priority of the test cases. The test case with the highest similarity is ranked first. Many kinds of identifiers can be selected, for example, method signature and data structure in services. Nguyen et al. [2011b] chose *CM-I* and *CMR-I* as the identifiers. The former contains class, method signatures, and terms extracted from those signatures, while the latter contains not only identifiers from *CM-I* but also request content sent by the service composition during the execution of the test case. The IR-based techniques claimed to have the advantage of finding profile-specific faults (i.e., faults affecting only users that have a specific location) over coverage-based techniques.

Other prioritization techniques. Another five techniques (P44–P48) were introduced for location-based services (LBSs), a kind of service enhanced with positional data [Dhar and Varshney 2011] and widely applied in many fields. LBS-enabled applications can augment the relevance of location-based outputs and avoid excessive outputs of irrelevant information. Zhai et al. [2014] proposed a set of black-box input-guided and POI-aware test case prioritization techniques for LBS-enabled applications. The input-guided techniques aim at maximizing the cumulated amount of uncertainty in the permutated test cases, while the POI-aware techniques aim at minimizing the cumulated amount of central tendency of the expected outputs of the permuted test suites to their corresponding inputs. They claimed that POI-aware techniques are more

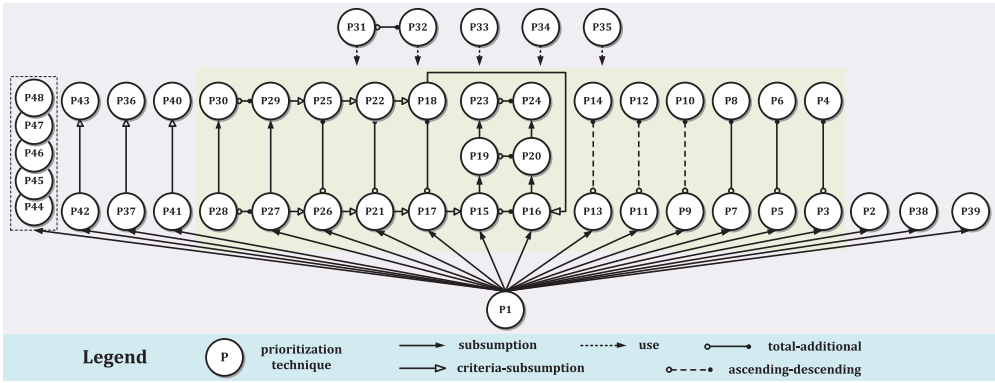

Fig. 4. Relations between different prioritization techniques.

effective and more stable than input-guided techniques for the majority of scenarios. Certainly, both techniques are more effective than the random ordering of test cases.

Prioritization techniques are related. It is valuable to identify some relationship among all 48 prioritization techniques. Given two test case prioritization techniques X and Y, we identify five relationships between X and Y.

—*Subsumption*. X subsumes Y if and only if any permutation of test suite produced by Y can also be produced by X [Mei et al. 2009]. For instance, we can prove that P15 subsumes P19. This is because P19 is the refined technique of P15, especially in the situation that test cases may have the same coverage for a given criterion. P19 ordered test cases by additional rule, while P15 did it randomly. Thus, the permutation of test cases generated by P19 could also be generated by P15.

—*Criteria subsumption*. X criteria subsume Y if and only if the criteria adopted by Y are subsumed in the criteria adopted by X. For instance, P17 criteria subsumes P15 since the criterion in P15 (WB) is a subset of criteria in P17 (WB and XRGB).

—*Total-additional*. X and Y have a total-additional relation if and only if (1) the coverage criteria of X are the same as those of Y, (2) X is a total-based strategy, and (3) Y is an additional-based strategy. For instance, P3 and P4 have a total-additional relation since both of them adopt coverage of SA as a criterion. P3 prioritized test cases independently with no consideration of effect caused by prioritized test cases, while P4 did it iteratively and synthetically by considering the coverage information of higher-priority test cases.

—*Ascending-descending*. X and Y have an ascending-descending relation if and only if (1) the coverage criteria of X are the same as those of Y, (2) X orders test cases in the ascending order of their coverage, and (3) Y orders test cases in the descending order of their coverage. For instance, P11 and P12 have an ascending-descending relation since both of them adopt coverage of WSDL tags as a criterion and P11 prioritized test cases in ascending order while P12 did so in descending order.

—*Use*. X uses Y if and only if the execution of Y is a necessary process of the execution of X. As mentioned earlier, framework prioritization techniques P31–P35 could use any concrete prioritization techniques from P3–P30.

Figure 4 shows the five types of relationships between 48 prioritization techniques. It is noted that the technique connected with a hollow circle by solid line is total based and that connected with a solid circle is additional based in a total-additional relation. The technique connected with a hollow circle by dashed line is ascending based and that connected with a solid circle is descending based in an ascending-descending relation.

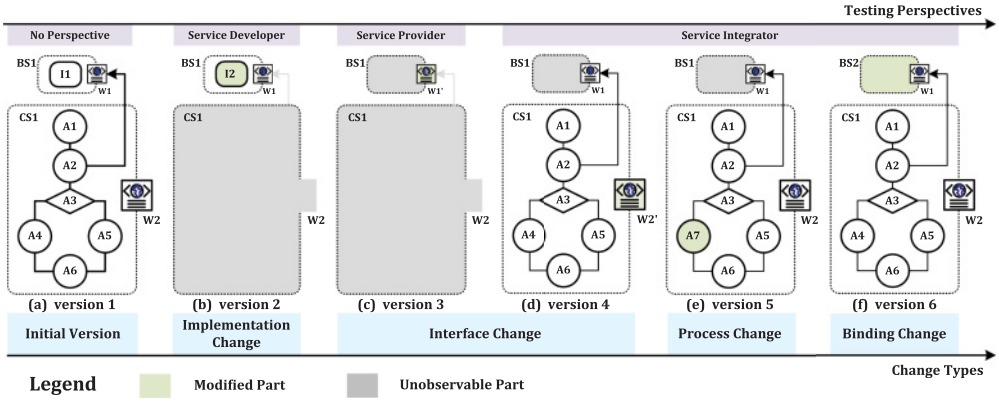

Fig. 5. Graphical representation of change types in the universal framework.

Since P44–P48 are all techniques for location-based service, we group them together in a dashed box for saving space.

4.4.2. Test Case Selection. The majority of test case selection techniques are modification aware [Yoo and Harman 2012]. This indicates that the work of test case selection requires identification of the modified parts of the service. Regarding focus question FQ2 (Which techniques are used to perform test case selection?), it is necessary to identify specific kinds of modification that may occur in web services since the change types in web services can serve as a guideline for choosing proper test case selection techniques. Unfortunately, the classification criteria in different primary studies, although they seem to be similar, are distinct. To minimize the misunderstanding, we try to synthesize the current classifications to create a universal framework to cover all possible change types in web services. Figure 5 depicts four change types with the help of the example illustrated earlier in Figure 3.

—*Implementation Change (ImC).* This is a change type for basic services, only observable from the perspective of service developers. It denotes the modification of concrete implementation of services, with no violation of the service interfaces. Consider Figure 5(b) as an example; service of version 1 (*V1* for short) evolved to *V2* since the implementation *I1* of *BS1* changed to *I2*. ImC originates from *modification of a component's specification* in Tarhini et al. [2006].
—*Interface Change (IC).* This is a change type for both basic services and composite services, observable from the perspective of both service providers and service integrators. For basic services, it denotes the modification of their interfaces; for composite services, it involves interface modification of both integrated services and partner services. Consider Figure 5(c)–(d) as an example; service *V1* evolved to *V3* since the interface of partner service *BS1* changed from *W1* to *W1'*; service *V1* evolved to *V4* since the interface of composite service *CS1* changed from *W2* to *W2'*. IC originates from *interface change* in Li et al. [2012].
—*Process Change (PC).* This is the change type for composite services, only observable from the perspective of service integrators. It denotes the modification of the structure of composite services, including addition or deletion of the partner services in process, modification of activities, and modification of execution order of activities. Generally, such change happens due to the modification of the service functional requirements. Consider Figure 5(e) as an example; service *V1* evolved to *V5* by

Table IX. Comparison of Test Case Selection Techniques

Ref.	Source	SUT	Method				Safety
			Input	Model	Approach	Granularity	
T1	S12	CWS	BPEL	BFG	Path Analysis	PC	
T2	S10	CWS	BPEL	BFG	Path Analysis	PC	
T3	S11	CWS	BPEL,WSDL	XBFG	Path Analysis	PC,IC,BC	
T4	S22	Java CWS	WSDL	JIG	Graph Walk	ImC	Safe
T5	S23	CWS	CFGs	Global CFG	Graph Walk	ImC,PC	Safe
T6	S24	CWS	CFGs	Global CFG	Graph Walk	ImC,PC	Safe
T7	S25	CWS	CFGs	Global CFG	Graph Walk	ImC,PC	Safe
T8	S26	CWS	CFGs	Global CFG	Graph Walk	ImC,PC	Safe
T9	S27	CWS	WSDL	TLTS	Modification Based	ImC,IC,BC	Safe
T10	S13	CWS	N/S	TGTS	Dependency Based	ImC,IC,PC	Safe

changing the activity *A4* to *A7*. PC originates from *process change* in Li et al. [2012] and *modification of the web application specification* in Tarhini et al. [2006].

—*Binding Change (BC).* This is also a change type for composite services, only observable from the perspective of service integrators. It denotes the replacement of a partner service by another candidate service with the same functionality. Different from PC in that the service integrator replaces the partner service for the purpose of changing the functionality to meet the modified requirements, BC may happen dynamically without notifying service integrators when certain nonfunctional properties are violated. Consider Figure 5(f) as an example; service *V1* evolved to *V6* since the partner service interacting with *A2* in *CS1* changed from *BS1* to *BS2*. BC originates from *binding change* in Li et al. [2012], *service composition/binding change* in Di Penta et al. [2007], and *connecting to the new web service* in Tarhini et al. [2006].

We identify 10 test case selection techniques from the primary studies. To better answer FQ2, we searched for several properties to compare the extracted test selection techniques, including the service type under test (*SUT*), concrete selection methods (*input*, *model*, *approach,* and service change *granularity* that can be detected), and *safety* of the techniques. Table IX lists this information.

Regarding the SUT, all techniques were dedicated for composite web services (CWSs). No technique aimed at basic services since black-box techniques for traditional software artifacts can be applied.

Regarding the input of the selection method, three techniques (T1–T3) concentrated on BPEL-based CWSs, where BPEL was used for depicting the structure of the CWS. WSDL is also required as the supplement in T3 to cover more types of service changes. Three techniques (T4, T9, and T10) used only interface information as input since the process structure of the CWS was not prerequisite. Among them, T4 and T9 took WSDL as input, while T10 did not mention any concrete interface implementation. Four techniques (T5–T8) required control flow graphs (CFGs) of all participating services as input.

Regarding the models generated for the selection method, CFG and CFG-based extension are widely employed. In T5–T8, CFGs of participating services were required to construct the global CFG of CWS. When modification occurred in CWS, it could be detected easily since corresponding change was reflected in the updated version of global CFG. To better depict the structure of BPEL, two customized CFG-based models were proposed in T1–T4. One is the BPEL Flow Graph (BFG), an extension of CFG to represent the process structure of a BPEL program. Afterwards, eXtensible BFG (XBFG) was proposed in T3, which modeled not only the process of BPEL programs but also the interactions between the process and its partner services. The other one is Java Interclass Graph (JIG) applied in T4, which was a CFG extension for Java

programs, first proposed in Harrold et al. [2001]. It requires the input of WSDL to be transformed to a local Java program that simulate related functionalities and behaviors. For the other two techniques, T9 use the Timed Labeled Transition System (TLTS) to depict CWS where internal behaviors of the participating services were also included; T10 modeled the CWS by using the Typed Graph Transformation System (TGTS) that concentrated on the data types, signatures, and rules between signatures.

Regarding the approaches, several papers have provided classifications of test selection techniques. We adopt the classification schema in Rothermel and Harrold [1996] since it covered more test selection techniques than the schema in Graves et al. [2001]. As indicated in Table IX, three out of 10 (30%, T1–T3) techniques used *path analysis techniques* [Benedusi et al. 1988]. These techniques took the set of service paths in both old version S and new version S' as input, which were expressed as BFG (XBFG) paths generated from the BFG (XBFG) model, and compared the paths from S and S' to identify paths as new, modified, deleted, or unmodified. They then analyzed test cases to determine the specific BFG (XBFG) paths traversed in S and selected all test cases that traversed modified paths. In particular, T1 focused on identifying modifications in BPEL concurrent control structures. However, it had the limitation of not being able to identify the cases of newly added structural activities in the process. T2 overcame this limitation. T3 further completed previous work and made it capable of handling binding changes in web service evolution.

Five out of 10 (50%, T4–T8) techniques used the *graph walk technique*. T4 proposed a safe regression test selection for Java-based web services based on Harrold's method [Harrold et al. 2001]. Similarly, T5 applied Rothermel's method [Rothermel and Harrold 1997] to general composite services. Both T4 and T5 took global CFGs of S and S' as their input, identified the dangerous edges by comparing their versions of CFGs, and selected test cases to be rerun from the test suite [Ruth et al. 2006; Ruth and Tu 2007c]. T6 improved the approach by considering the three possible scenarios of concurrent changes on services to ensure the test consistency. T7 and T8 further perfected the approach, considering the privacy issue of shared services during the test selection process since T4 and T5 might require sensitive implementation details of participating services (such as the CFGs) that service providers do not want to share. Hence, T7 used several privacy-preserving techniques to protect the sensitive information involved in CFGs of shared services, with no negative impact on the overall effectiveness of the approach [Ruth 2011]. In contrast, sensitive details of participating services were not required at all in T8 since it used only locally available information.

In addition, T9 used the *modification-based technique* that identified modifications from generated TLTS and selected test cases that would cover the modified part. This approach can also generate test cases for the new and existing services and operations. T10 used the *dependency-based technique* that selected test cases based on an analysis of the dependencies and conflicts between visual contracts specifying the preconditions and effects of operations.

Regarding the granularity, most techniques supported identifying and handling PC and ImC. Certainly, detecting ImC largely depends on whether the implementations of the participating services are available. Three (T3, T9, and T10) techniques supported detecting IC since the other techniques did not include the interface information for analysis. Only two techniques (T3 and T9) supported BC.

A key property to assess the regression test selection techniques is whether they are *safe* or not. With a safe technique, every test case from the original test suite that can expose faults in the modified program is still selected [Rothermel and Harrold 1997]. That is, all faults found with the full test suite are also found with the test cases picked by the test selection technique [Engström et al. 2010]. In our study, seven techniques (T4–T10) were safe.

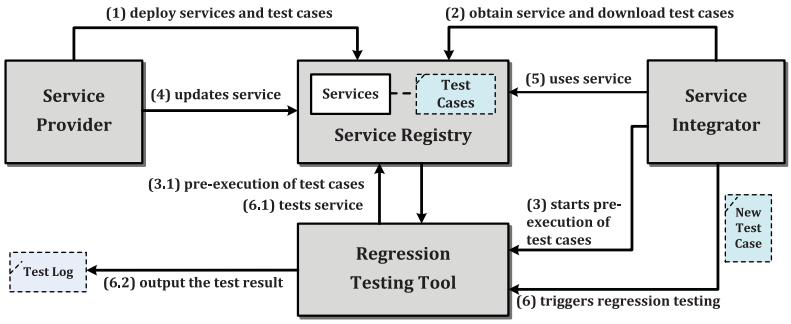

Fig. 6. Collaborative regression testing of web service.

4.4.3. Test Suite Minimization. Test suite minimization techniques are less popular than the other two branches of techniques since only two primary studies concentrate on this topic. A possible reason is that minimization is often integrated into the process of test case generation as an optimization step. Regarding the focus question FQ3 (Which techniques are used to reduce the size of test suite in web service?), only studies S5 and S8 can provide the answer. S5 proposed a novel cost-aware optimal test suite minimization to reduce the runtime testing cost. The approach calculated three measurements—coverage, reliability, and execution cost of each test case—and adopted multiobjective optimization to reduce test cases. S8 proposed a reduction technique based on pairwise combination of input parameters. BPEL-based web service composition was modeled by High-level Petri Nets (HPNs). After translation, the equivalent HPN could be verified with an existing mature tool and then the size of the test suite could be reduced by applying the reduction rules [Dong 2008]. Overall, test case minimization has attracted less attention than others.

4.4.4. Others. Apart from the previous three mainstream branches of regression testing techniques, primary study S7 proposed a *collaborative* regression testing technique that covered both functional and nonfunctional changes of web service by involving different stakeholders, including service providers, service registries, and service integrators, to join the whole testing process, with a clear assignment of responsibilities and duties. Figure 6 shows the collaborative process between the stakeholders, which is adopted from Di Penta et al. [2007] with simplification of some optimization policies.

The key issue is to improve the testing efficiency. The collaborative process required stakeholders to share their test cases and execution histories for reuse. Test cases associated with the web services were uploaded simultaneously to the service registry. Moreover, execution traces of the test case were also monitored and recorded in a public database. When other stakeholders intend to test the services, they do not need to generate the existing test cases and run them again if their execution histories were recorded in the database.

Another important issue is when to perform regression testing if no notification was received by the service integrator. Three time points were put forward to deal with different situations: (1) triggered by service version changes: this required that version information be inserted into the service interface so that the service integrator could detect the change; (2) periodic retesting; this required a regression testing tool to automatically perform testing periodically; (3) whenever the service was used: this required executing test cases before each invocation of the service.

4.5. Validation

All new regression testing approaches should be validated. To answer research question GQ5 (What method is being used to validate the research?), we consider two perspectives in turn. First, a classification of different validation methods adopted by primary studies is presented in Section 4.5.1. All web services involved in the validation process are discussed in Section 4.5.2, with corresponding information of scale, accessibility, source availability, category, and usage. Moreover, a summary of how the primary studies use these services is also included in this subsection.

4.5.1. Types of Validation. We adopt the classification of validation methods proposed by Shaw [2003], which is widely used in software engineering to assess the validity of approaches. There are six categories of validation methods listed as follows:

—*Analysis*. Rigorous derivation and proof or carefully designed experiment with statistically significant results.
—*Experience*. The approach has been applied in real-world scenarios or projects and the evidence on correctness/usefulness/effectiveness collected.
—*Evaluation*. A set of examples is used to illustrate the proposed approach, with an assessment of stated criteria on the gathered information from the execution of examples.
—*Example*. A few small-scale examples are used to illustrate the proposed approach, without any evaluation or comparison of the execution result.
—*Persuasion*. An abbreviated text description is presented to convince readers that the approach is useful.
—*Blatant Assertion*. No serious attention is paid to validation.

Table X shows the distribution of validation methods applied to the primary studies. We found nine studies using *analysis* where theorem proof was applied in one study and carefully designed experiments in eight studies. In Dong's work, three theorems were proved to show that the reduction technique based on pairwise combination was effective [Dong 2008]. Eight studies selected some frequently used benchmark services to compare the effectiveness of the proposed regression prioritization strategy with other existing technologies [Hou et al. 2008; Mei et al. 2009; Mei et al. 2009, 2011, 2012, 2013a, 2013b; Zhai et al. 2014].

Twelve studies evaluated the regression testing approaches using assessment criteria based on the collected execution information, while another seven studies worked with simplified examples to illustrate their approaches. These studies are represented in the *Evaluation* and *Example* rows, respectively.

Finally, only two primary studies used *persuasion* as the validation method [Ruth 2008; Bai and Kenett 2009].

From the results, *evaluation* is the most used validation method, while *analysis* and *example* are also widely applied. It should be noted that *experience* is not applied in any studies. A possible reason is that most large-scale services, especially cross-organization service [Ye et al. 2009], are not open, which makes it difficult to access the implementation of these industry services.

4.5.2. Types of Services. As mentioned in Section 4.5.1, most primary studies (90%, 27/30) used *analysis*, *evaluation,* or *example* to validate their proposed approaches where demonstrative *services* are indispensable parts of these validation. Hence, it is necessary to anatomize the usage of selected web services in the validation process. For this reason, we try to extract all services used in primary studies and obtain their essential information, including their names, scale accessibility over the network, and source code availability. Also, we categorize all services into five kinds, *ad hoc*, *sample*,

Table X. Distribution of Primary Studies by Validation Methods

ValidationMethod	Primary Studies																														
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	**Total**
Analysis								✓	✓					✓	✓	✓	✓	✓	✓											✓	**9**
Experience																															**0**
Evaluation	✓	✓			✓		✓						✓							✓	✓		✓		✓	✓		✓	✓		**12**
Example			✓			✓				✓	✓	✓										✓					✓				**7**
Persuasion				✓																				✓							**2**
Blatant Assertion																															**0**

Table XI. Overview of Web Services Collected from Primary Studies

ID	Service Name	Scale	Accessibility	Availability	Category	# of usage
WS1	Weather Monitoring	15A	✓	×	public	2
WS2	BibleWebservice	6A	✓	×	public	1
WS3	Airline Reservation	8A	×	×	ad hoc	1
WS4	ATM Example	12A	×	×	ad hoc	1
WS5	Dnsjava	-	×	✓	wrapped	1
WS6	HotelService	-	×	×	sample	1
WS7	RestaurantService	-	×	×	sample	1
WS8	Travel Arrange	3PS	×	×	ad hoc	1
WS9	Travel Agent System	12S	×	×	ad hoc	1
WS10	Loan Flow	3PS	×	○	sample	1
WS11	Bug Tracker	15S	×	✓	wrapped	1
WS12	ATM	180L	×	○	sample	6
WS13	Gymlocker	52L	×	○	sample	6
WS14	Loanapproval	102L	×	○	sample	7
WS15	Marketplace	68L	×	○	sample	6
WS16	Purchase	125L	×	○	sample	6
WS17	Triphandling	170L	×	○	sample	6
WS18	Buybook	532L	×	○	sample	6
WS19	Dslservice	123L	×	✓	sample	6
WS20	eBayfinder	20*k*L	×	×	ad hoc	2
WS21	Purchase Order	4S	×	×	ad hoc	4
WS22	Loan Application	9S	×	×	ad hoc	1
WS23	Loan Brokerage	5S	×	×	ad hoc	1
WS24	SCM System 1	15S	×	×	ad hoc	1
WS25	SCM System 2	8S	×	×	ad hoc	1
WS26	Travel Agency	4S	×	×	ad hoc	1
WS27	Stock-Buy-Sell	-	×	×	ad hoc	1
WS28	City Guide	3.3*k*L	×	×	ad hoc	1
WS29	Complex	-	×	×	wrapped	1
WS30	Shipping Workflow	14PS	×	×	ad hoc	1

wrapped, *public* and *real scenario*, to provide clues of how primary studies use these services to validate their approach. The classification of services is shown as follows:

—*Ad hoc* services are self-designed by the authors just for illustrating the proposed approach.
—*Sample* services are attached in the specification of protocols (standards) or tool packages. They help the user to better understand the content of protocols (standards) or quickly start using the tools.
—*Wrapped* services are derived from existing traditional software systems and are wrapped by web service technology.
—*Public* services reside in a public server that can be accessed from the Internet.
—*Real scenario* services are industrial grade and widely used in the large-scale projects.

Table XI lists the detailed information of 30 services collected from 30 primary studies (12 services in study S10 and 60 services in S29 are not included since no concrete information is available from these two studies). The first two columns show the ID and name of services, respectively. The third column provides the scale of service according to four different measures mentioned in primary studies. *PS* represents the number of partner services involved in the composite service; *S* represents the number of total services involved; *A* represents the number of activities in the process; *L* represents the LOC (Line of Code) of the service definition. The *accessibility* in the fourth column shows whether the services can be accessed (invoked) over the network.

The *availability* in the fifth column shows whether the source code of the services can be obtained. Symbol "✓" represents that the services (source codes of services) can be accessed over the network (gained from specified resource); "×" represents the opposite result of "✓"; "○" represents that the source codes of the services were available but now unavailable. The sixth column lists the category of each service. The last column summarizes the number of usage for each service in all primary studies. It reflects the share usage of services.

Next, we analyze the usage of services in corresponding primary studies. Table XII provides this information where three key aspects are considered: types of services used, number of services analyzed, and number of total versions of selected services *w.r.t.* each service listed in Table XI. About half of the studies (46.7%, 14/30) used *ad hoc* services, and most of them were not shared and reused in the other related work. Seven studies used *sampled* services where most were extracted from service-related open-source tools. *Wrapped* services and *public* services were adopted in three and two primary studies, respectively. No study used *real scenario* services. Two studies did not mention the services that were used in their papers.

Considering the test scale in validation, most studies (66.7%, 20/30) chose no more than five services, and many as few as one service. Only two studies used more than 10 services, but the concrete information of these services was not provided [Li et al. 2008; Tsai et al. 2009]. Most services in primary studies that adopted *example* in validation were applied to illustrate the feasibility of the approaches (e.g., Chen et al. [2010]). Services in primary studies that adopted *analysis* or *evaluation* were further applied to evaluate the correctness and effectiveness (e.g., Mei et al. [2009]).

Regarding the version issue, the number of selected versions for each service basically varies between two and six. A possible reason for this low number is that most evolutive versions are constructed manually to satisfy the requirement of validation, which makes it difficult to generate more versions. Zhai et al. [2014] used 35 faulty versions to evaluate their approach where all versions are generated automatically using mutation operators. There are six studies that adopted only one version per service in validation. It is acceptable since multiversions are not required in the validation of test case prioritization and test suite reduction techniques.

From Table XI and Table XII, we have the following observations:

(1) Most services used in primary studies were unavailable or inaccessible through the Internet. Looking at the third and fourth columns in Table XI, almost all (93.3%, 28/30) services could not be accessed over the network. Only WS1 and WS2 in *WebserviceX*[10] could be invoked directly. Besides, many (63.3%, 19/30) services could not be obtained since the authors did not share these resources. The source code of only three services could be downloaded from the specified websites. In particular, WS12 is the *sample* service from the *web service invocation framework*.[11] WS11 and WS18 are *wrapped* services from *DNSJava*[12] and *Bug Tracking System*,[13] respectively. It is noted that eight services labeled by ○ were previously available but now unavailable because the BPEL repository from IBM[14] (supplying the services WS12–WS18) and the project of BPEL Process Manager[15] (supplying the service WS10) have been retired.

[10]http://www.webservicex.net/.
[11]http://ws.apache.org/wsif/wsif_samples/.
[12]http://www.xbill.org/dnsjava/.
[13]http://btsys.sourceforge.net/.
[14]http://www.alphaworks.ibm.com/tech/bpelrepository.
[15]http://www.oracle.com/technology/products/ias/bpel/.

Table XII. Distribution of Primary Studies by Validation Services

		Primary Studies																													
		S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	**Total**
ServiceUsage	Ad-hoc			✓		✓	✓		✓	✓		✓	✓								✓	✓	✓	✓				✓	✓		✓	**14**
	Wrapped					✓		✓						✓																		**3**
	Sample							✓							✓	✓	✓	✓	✓	✓												**7**
	Public	✓	✓																													**2**
	Real-scenario																															**0**
	N/S										✓																			✓		**2**
# of services		1	2	1	-	2	1	3	1	1	12	1	1	1	8	8	8	8	8	8	1	1	1	5	-	1	1	1	60	60	1	-
# of versions		1	2	2	-	2	1	11	1	1	-	4	2	3	60	60	60	60	43	43	6	6	-	-	-	-	-	4	-	-	35	-

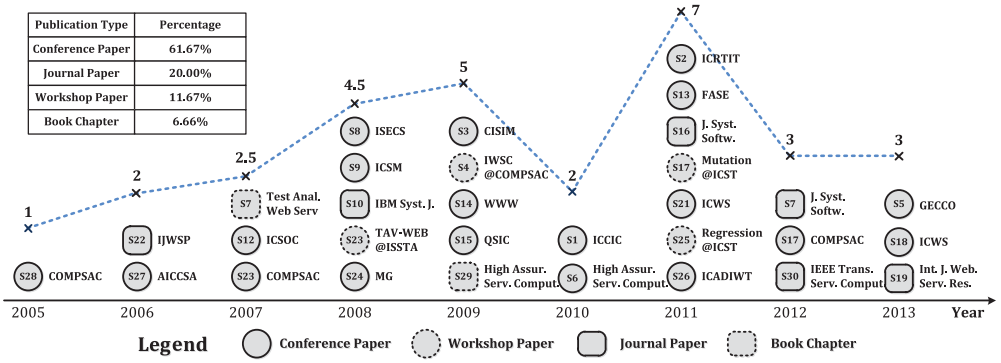

Fig. 7. Publication chronology (Since primary study S23 is composed of Ruth et al. [2007] published in 2007 and Ruth and Tu [2008] in 2008, two halves are added to the publication number in 2007 and 2008, respectively.).

(2) The metrics used for the service scale were inconsistent across primary studies. Seven studies used LOC to measure service scale, which was a common size metric at the code level. Four studies adopted the number of activities, which focused on the structure and flow relation, to reflect the internal complexity of the service. In addition, seven studies used the number of involved services (or partner services) to show the scale of the services from the perspective of service integrator or provider. This inconformity made it hard to comprehend and compare the size of different services selected for validation.
(3) The scales of the services were not large. For example, regarding those services that employed LOC as the scale metric (WS12–WS20 and WS28), only services WS20 and WS28 had a scale of more than one *KLOC*. As validation with small-scale services could not fully evaluate the effectiveness of the approaches, the results provided by such studies might not be generalized to larger scales. A possible reason for this situation is the *high testing cost* of large-scale services and the lack of free open-source services. To avoid the charge of commercial services, researchers often design *ad hoc* services themselves or use free *public* services for validation.
(4) The sharing of services was very low. Only a small number of services were adopted by other studies. Also, the reused services were usually limited to the studies by the same research group. For example, services from WS12 to WS19 were reused in studies S14–S19, where all six studies were published by the same author. This might be due to the difficulty in finding services that match the requirements of specific studies and the lack of large-scale public services.

4.6. Distribution of Studies

We are also interested in other related information, such as *when* and *where* the primary studies were published. This provides a direct vision of the recent research trend of web service regression testing. We collect publication year and source for each primary study. In addition, we categorize all primary studies to four publication types: *conference paper*, *journal article*, *book chapter,* and *workshop paper*. Figure 7 provides a chronology for all primary studies. We put the abbreviation of conference (or journal, book) name as a representative of the source beside each study. For those workshop papers, *A@B* is used as the format of the source where A is the abbreviation of the workshop name and B is the name of the conference that holds the workshop. The collected data helps to answer research question SQ1 (How much activity about

the research of web service regression testing has there been in recent years?) and SQ2 (Where have the researches been published?).

The dashed line in Figure 7 summarizes the number of primary studies published per year from 2005 to 2013. Although the time period defined in our search strategy is between 2000 and 2013, no primary study was found until 2005. This may be because, on the one hand, the technology of web service emerged in the early 2000s but the need for testing techniques has not been recognized until recently [Palacios et al. 2011]; on the other hand, with the increasing service scales and corresponding testing cost, regression testing techniques are urgently required to improve testing efficiency and reduce testing cost. From 2005, around one to seven primary studies per year were found with a gradual upward trend. Not until 2011 did we find a marked increase in the number of primary studies (seven). This fact indicates that more and more effort is being dedicated to this important area of research. However, there was a relatively big decrease in 2012 and 2013, where only six primary studies were published in total. Consequently, more data is needed in the next few years to determine whether the research of this area has reached a peak.

The table in Figure 7 summarizes the percentage of primary studies according to the venue of publication. Among 30 primary studies, 61.7% (18.5 of 30) were published in conferences. Three primary studies (S17, S23, and S28) were published in *COMPSAC*, and two studies (S18 and S21) were published in ICWS. The rest of the studies were published in different conferences. Twenty percent (six of 30) were published in journals. Two papers (S11 and S16) were published in *JSS*. Finally, 11.7% (3.5 of 30) were published in workshops, where two of them were from the workshops associated with *ICST*.

From this collected data, we offer three observations:

(1) The number of papers on regression testing of web services is growing but still relatively small. However, given this specialized research area and a relatively short history, the number of primary studies can be considered significant. In addition, as there are still many challenges to be addressed, more research work may be in progress or published in future.
(2) Most papers were published in proceedings (20, including conference and workshops). This might be because many interesting methods have been proposed in recent years, but their theoretical frameworks are still not solid.
(3) Five primary studies (S10, S11, S16, S19, S30) appeared in reputable journals (*JSS*, *TSC*, *Int. J. Web. Serv. Res.,* and *IBM Syst. J.*). However, no papers were published in top journals in the software engineering area (e.g., TSE and *TOSEM*). Similarly, no papers were published in top conferences (e.g., ICSE and *FSE*), although some primary studies were published in some leading conferences (e.g., *FASE*) or the subfield conferences, like service-related conferences (e.g., *ICWS*, *ICSOC,* and *SOSE*) and maintenance-related conferences (e.g., *ICSM*). In addition, no primary studies have been published in top testing-related conferences (e.g., *ISSTA*, *ICST*), although three primary studies were published in workshops associated with these conferences. The lack of publications in top journals and conferences is probably due to the lack of elaborate evaluation of proposed approaches, and especially due to the small size of case studies.

5. DISCUSSION

5.1. Summary of the Mapping Study

Through a comprehensive analysis of 30 primary studies concentrating on the topic of web service regression testing, we find that many works have been presented in the

last 9 years. At the preliminary stage, most work focuses on how to import the existing traditional software regression testing techniques into web services. Afterwards, many studies focus on how to solve the challenges brought by the inherent characteristics of web services.

Under the scenario of web service regression testing, we classify the service stakeholders into five categories: service developers, service providers, service registries, service integrators, and service users. Resources managed by one stakeholder are often not available to other stakeholders, which affects the design and selection of regression testing approaches for web service. According to the statistics, service integrators play the most active role in testing activity as their tasks are more challenging, which requires them to take account of both the service process and partner services. Service developers, service providers, and service registries are also mentioned in some of the primary studies. Only service users are not regarded as indispensable participants in the testing process because they are not normally involved in the testing process.

Regarding the challenges, seven representative ones were mentioned in the primary studies: cost, autonomy, test timing, dynamics, concurrency, quota constraints, and privacy issues. Most of the studies emphasize the first two issues, discussing how to possibly reduce the test cost, especially without access to service implementation. Although research activities on the other five challenges are relatively low, they are quite significant, especially for large-scale complex services and services that operate across organizations. Certainly, the development of web services, including infrastructure and implementation, will likely bring more challenging problems.

Considering the system under test, 93.3% of the primary studies aimed at composite services and 26.7% of the studies aimed at basic services. In addition, six of the studies could process both basic and composite services. Regarding the standards/protocols, WSDL was the most widely used service interface description language for both basic service and composite service (63.3%). BPEL was widely acceptable for depicting the behavior of composite service (40%).

Regarding the regression testing techniques, 56.7% of the primary studies studied the test case prioritization techniques, while 33.3% of them studied test case selection techniques. Only two studies focused on the test suite minimization. It is noteworthy that there is one special study proposing a collaborative regression testing framework to invite all service stakeholders to test together. In detail, a total of 48 prioritization techniques are analyzed where most of them are coverage based (68.6%), FEP based (12.5%), and IR based (4.2%). Five of the prioritization techniques are specific for location-based services. A total of 10 test selection techniques were proposed to cover four possible change types in web service, including implementation change, interface change, process change, and binding change. Among these techniques, most are path analysis based and graph walk based and 60% of them claimed they were safe.

The validation methods in the primary studies had several limitations. Many studies (63.3%) still adopted *evaluation* and *example*, which are not as rigorous and convincing as *analysis* in assessing the validity of the approaches. In addition, two studies used *persuasion* and no study applied *experience* to illustrate the applicability of the approach.

Similarly, the services used in validation also had several limitations. First, most services (93.3%) could not be accessed and many services (63.3%) could not be obtained through the Internet, which might prevent others from repeating the same experiments or using similar examples in their own studies. Second, the metrics of service scale were not consistent across primary studies. It might become an obstacle for comparison of related works. Third, the scales of the services were not large as many of them (46.7%) were ad hoc examples. The lack of large-scale services, especially those extracted from

real scenario, made it difficult to fully validate the proposed techniques. Finally, the sharing of services was very low, which might be partly due to the first limitation.

5.2. Implications for Future Studies

This mapping study not only offers useful information for researchers who are interested in improving the quality of web service regression testing but also identifies gaps in this research topic.

First of all, the service integrator is the most active stakeholder while the service registry and service user are rarely involved in regression testing activities. However, testing from one single perspective usually has limitations. For example, service developers and providers cannot perform realistic nonfunctional testing as their testing is not under the actual service usage environment; service integrators have higher test costs due to frequent invocations of nonfree partner services during the testing process. Hence, the involvement of multiple stakeholders in collaborative regression testing, as suggested by Di Penta et al. [2007], might be an efficient strategy. Testing from multiple perspectives would maximize the sharing of code resources, test cases, and test processes to reduce test costs and improve test efficiency.

Most researchers focus on the regression testing of orchestration-based composite services, especially BPEL-based services. However, no study concentrates on choreography-based composite services. Hence, regression testing of choreography-based services (e.g., WS-CDL based services) requires more investigation. Besides, only one study considers regression testing of semantic web service, which suggests that it would also be fruitful to develop new techniques for semantic web services, such as OWL-based web services.

Coverage-based test case prioritization is one of the most frequently adopted prioritization techniques. Regarding the coverage strategies, total greedy and additional greedy algorithms are used separately to prioritize test cases. The combination of total and additional greedy search algorithms would be meaningful since Zhang et al. [2013] provide the evidence that many strategies combining the total and additional strategies are more effective than either of those strategies and other search algorithms. In addition, other algorithms, such as 2-optimal, hill climbing, and genetic algorithms [Li et al. 2007], may achieve a higher fault detection rate.

In the validation of the proposed techniques, *evaluation* and *example* are the most-used validation methods, which are not as good as a rigorous *analysis* and proof. Although *analysis* has been applied to a few primary studies, the scale of service samples is too small to confirm the general applicability of the studies. Besides, there is no common framework for comparing the regression testing techniques. Thus, it would be an important contribution to develop an assessment method, similar to that proposed by Rothermel and Harrold [1997] for test case selection techniques in traditional software testing. For those studies applying analysis as their validation method, additional experimentation using real scenarios or standard examples should be conducted to provide data for comparison with other related studies.

As many industrial leaders have promoted the use of service-oriented business processes to build their enterprise systems [Mei et al. 2009], applying web service regression testing techniques in industry is becoming significant and challenging. Since most techniques are only used and validated in small cases, collaborating with the industry and applying these techniques in real-world projects is the first imperative.

5.3. Related Work

Canfora and Di Penta [2006, 2009] provided a common scenario of regression testing process for web service, illustrated the challenges from the viewpoints of different stakeholders, and presented solutions for four levels of testing, including unit testing,

Table XIII. Comparison of Related Surveys

Survey Paper	Domain	Surveyed Techniques	# of common surveyed study
Rothermel and Harrold [1996]	Traditional Software	Test Case Selection	0
Rothermel et al. [2001]	Traditional Software	Test Case Prioritization	0
Elbaum et al. [2002]	Traditional Software	Test Case Prioritization	0
Rothermel et al.[2002]	Traditional Software	Test Suite Reduction	0
Canfora and Di Penta [2009]	Web Service	Unit/Integration/Regression/ Nonfunction Testing	3
Zakaria et al. [2009]	BPEL-based Service	Unit Testing	2
Bozkurt et al. [2013]	Web Service	All Kinds of Testing	12
Engström et al. [2010]	Tradition Software	Test Case Selection	0
Palacios et al. [2011]	SOA with Dynamic Binding	All Kinds of Testing	0
Yoo and Harman [2012]	Traditional Software	Regression Testing	2
Our Paper	**Web Service**	**Regression Testing**	-

integration testing, regression testing, and nonfunctional testing. Although they covered the topic of web service regression testing, the set of publications in their survey was incomplete and a comprehensive comparison was not presented. As an extension of Canfora and Di Penta's paper, Bozkurt et al. [2013] focused on the functional aspects of regression testing and classified the research according to the concrete techniques. They also allocated a section to summarize regression testing techniques. However, most techniques mentioned were test selection techniques. In addition, comparison between these techniques was not presented.

There are some surveys on testing web service using a mapping study or systematic review. Palacios et al. [2011] presented a mapping study of testing in service-oriented architecture with dynamic binding. The main difference is that their review restricted the scope to the SOA with dynamic binding, whereas our review aims at identifying research on regression testing techniques in web services, with both static binding and dynamic binding. In addition, Zakaria et al. [2009] presented a systematic review that analyzed and evaluated 27 different studies on unit testing approaches for BPEL-based services, concentrating on unit testing approaches, test case generation, test coverage criteria, empirical evidence, and current study trends. Notably, two regression testing techniques in their studies (i.e., Liu et al. [2007], and Wang et al. [2008]) are also included in our survey. The main difference is that their survey studied all unit testing techniques on services of a specific type, while our study focuses on one specific branch of testing techniques.

Many reviews were published in the topic of traditional software regression testing techniques [Yoo and Harman 2012], including the reviews on three mainstream branches, that is, test case prioritization [Rothermel et al. 2001; Elbaum et al. 2002], test case selection [Rothermel and Harrold 1996; Engström et al. 2010], and test suite reduction [Rothermel et al. 2002]. We adopted this classification schema to analyze regression testing techniques in web services. In addition, Yoo and Harman [2012] discussed open problems and potential directions in web service and claimed that regression testing of web service and SOA contained many challenging problems. Our study comprehensively summarizes the overall challenges and presents the current research state. We also list some promising directions for further study.

Table XIII lists all the related surveys according to their publication year and compares their scopes from the perspective of application domain and surveyed techniques. At the last column, we count the common surveyed studies in each related survey papers *w.r.t.* our article. It shows that only one paper [Bozkurt et al. 2013] has more than five surveyed studies in common with our article. More than half of the primary studies

in our survey have never been reviewed in the related works, which strongly supports the need for our article.

5.4. Limitation of this Review

There are three potential limitations of our mapping study: (1) bias in the publication selection, (2) inaccuracy in data extraction, and (3) specific usage of methodology in the survey process.

First, to ensure that the process of selection was unbiased, we designed a search strategy in advance that divided the research questions into three categories. Based on these questions, we set keywords and search terms that would enable us to identify the relevant literature. However, the keyword selection was subjective, which might result in missing some relevant studies. Furthermore, an all-round and significant group of electronic databases focusing on computer science was selected as the search scope for covering as many relevant studies as possible. Nevertheless, it is possible that some related studies, which should be included in our study, may have been omitted if they are not within our defined search scope.

Second, some inaccuracy of extraction result is possible. In the pilot of the data extraction process, we extracted the detailed information of each primary study with a predefined extraction form (in Appendix D). However, we often found that the data extraction process was incomplete because of the lack of sufficient information in some of the studies. More specifically, we frequently found that techniques or approaches were not depicted adequately, with the issues of validation methods not always addressed; that the analysis for experimental results was not explained well; and that samples and study settings were not presented clearly. This impacts the accuracy of extraction results.

For our review, we use mapping study [Kitchenham and Charters 2007]. During the survey process, we also import classification of stakeholders in Canfora and Di Penta [2006] and validation types summarized in Shaw [2003] to classify the primary studies. The usage of such a specific methodology, classification, and standard may also impact the survey results.

6. CONCLUSIONS

A mapping study has been performed on the topic of web service regression testing. We did a large-scale literature search on 11 electronic databases and a total of 30 papers have been selected as primary studies. Regarding the *general* questions, we have identified that:

*GQ*1: Four out of five service stakeholders are involved in the regression testing activities, namely, service developer, service provider, service registry, and service integrator. Among them, service integrator is the most attractive perspective from which most research work starts. Testing from this perspective also induces more challenges to conquer than others.

*GQ*2: Many inherent and specific characteristics of web services, such as loose coupling and ultra-late binding mechanism, bring seven significant challenges. Among all primary studies, *test cost* and *autonomy* issues are hotspots and several approaches have been proposed to solve those issues. Besides, the other five challenges, *test timing*, *dynamics*, *concurrency*, *quota constraints,* and *privacy* issues, also deserve further research.

*GQ*3: For the SUT, most emphasis has been put on composite services instead of basic services. Regarding the standards used in SUT, WSDL is popular in specifying interfaces of web services, with most regression testing techniques relying on

it. Considering the composite services, orchestration-based services, especially those specified in BPEL, have been largely studied. However, less attention has been paid to regression testing of either choreography-based services or semantic-based services.

*GQ*4: Test case prioritization, selection, and minimization are three major regression testing techniques applied to the web service. In addition, a collaborative regression testing approach has been proposed to encourage more service stakeholders to join in the testing activities, for the purpose of reducing the test cost and improving the completeness of testing.

*GQ*5: Regarding the validation method applied in primary studies, *evaluation*, *analysis,* and *example* are most widely used. Considering the services employed in these validation methods, most of them are *ad hoc* and *sample* and the scales are relatively small. In addition, the share rate of these services across studies is quite low since most of them are unavailable or inaccessible through the Internet.

Regarding the *focus* questions, we have identified that:

*FQ*1: A total of 48 prioritization techniques from 17 primary studies have been analyzed in this survey. Among them, coverage-based, FEP-based, and IR-based approaches have been adopted to prioritize test cases. In addition, five techniques aim at services of specific domain (i.e., location-based services).

*FQ*2: By comparing the classifications of change types in web services from all primary studies, we synthesize them into a unified category, including *implementation change*, *interface change*, *process change,* and *binding change*. Ten test selection techniques from 10 primary studies have been proposed, but no technique can cover all four change types. Among them, path-analysis-based and graph-walk-based techniques are widely adopted to select test cases. Sixty percent claim that their approaches are safe.

*FQ*3: Only two studies focus on the test case minimization techniques.

We have also found that:

*SQ*1: The primary studies in this field start from 2005. During the past 9 years, the number of publications each year varies from one to seven, with a gradual upward trend.

*SQ*2: Most primary studies are published in proceedings, and some studies are published in reputable journals. The lack of publications in top journals and conferences indicates that web service regression testing is still an emerging topic.

Overall, current research on web service regression testing has produced quite a few solid works. However, there are still many big challenges in this area, such as privacy, dynamics, and concurrency issues. We encourage all the researchers to dedicate more effort to tackle the many interesting problems.

APPENDIX

A. SELECTED ELECTRONIC DATABASES

The selected electronic databases used in the primary study are shown in Table XIV.

B. SELECTION OF STUDIES IN THE SAME RESEARCH LINE

The process of selecting the most representative primary study in the same research line is shown as follows:

Table XIV. Selected Electronic Databases

Electronic Databases	Website
ACM Digital Library	http://portal.acm.org/
Citeseer Library	http://citeseer.ist.psu.edu/
EI Compendex	http://www.engineeringvillage.org/
Google Scholar	http://scholar.google.com/
IEEE Xplore	http://ieeexplore.ieee.org/
Inspec	http://inspecdirect.theiet.org/
ScienceDirect	http://www.sciencedirect.com/
DBLP Bibliography	http://dblp.uni-trier.de/
SpringerLink	http://www.springerlink.com/
Web of Science	http://www.isiknowledge.com/
Wiley online Library	http://onlinelibrary.wiley.com/

Table XV. Quality Assessment for Primary Studies

Primary Study	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Askarunisa et al. [2010]	✓	✓	✓	✓	✓	✓	✓	✓
Askarunisa et al. [2011]	✓	✓	✓	✓	✓	✓	✓	✓
Athira and Samuel [2010]	✓	✓	✓	✓	✓	✓	✓	✓
Bai and Kenett [2009]	✓	✓	✓	✓	✓	✓	✓	✓
Bozkurt [2013]	✓	✓	✓	✓	✓	✓	✓	✓
Chen et al. [2010]	✓	✓	✓	✓	✓	✓	✓	✓
Di Penta et al. [2007]	✓	✓	✓	✓	✓	✓	✓	✓
Dong [2008]	✓	✓	×	✓	✓	✓	✓	✓
Hou et al. [2008]	✓	✓	✓	✓	✓	✓	✓	✓
Li et al. [2008]	✓	✓	✓	✓	✓	✓	✓	✓
Li et al. [2012]	✓	✓	✓	✓	✓	✓	✓	✓
Liu et al. [2007]	✓	✓	✓	✓	✓	✓	✓	×
Khan and Heckel [2011]	✓	✓	✓	✓	✓	✓	✓	✓
Mei et al. [2009]	✓	✓	✓	✓	✓	✓	✓	✓
Mei et al. [2009]	✓	✓	✓	✓	✓	✓	✓	✓
Mei et al. [2011]	✓	✓	✓	✓	✓	✓	✓	✓
Mei et al. [2012]	✓	✓	✓	✓	✓	✓	✓	✓
Mei et al. [2013a]	✓	✓	✓	✓	✓	✓	✓	✓
Mei et al. [2013b]	✓	✓	✓	✓	✓	✓	✓	✓
Nguyen et al. [2011a]	✓	✓	✓	✓	✓	✓	✓	✓
Nguyen et al. [2011b]	✓	✓	✓	✓	✓	✓	✓	✓
Ruth et al. [2006]	✓	✓	✓	✓	✓	✓	✓	✓
Ruth et al. [2007]	✓	✓	✓	✓	✓	✓	✓	✓
Ruth [2008]	✓	✓	✓	✓	✓	✓	✓	✓
Ruth and Rayford [2011]	✓	✓	✓	✓	✓	✓	✓	✓
Ruth [2011]	✓	✓	✓	✓	✓	✓	✓	✓
Tarhini et al. [2006]	✓	✓	✓	✓	✓	✓	✓	✓
Tsai et al. [2005]	✓	✓	✓	✓	✓	✓	✓	✓
Tsai et al. [2009]	✓	✓	✓	✓	✓	✓	✓	×
Zhai et al. [2014]	✓	✓	✓	✓	✓	✓	✓	✓

—Li et al. [2012, 2010] and Wang et al. [2008]. Li et al. [2012] is the journal version of Li et al. [2010] and Wang et al. [2008], so *Li et al. [2012]* is selected as the primary study.
—Tsai et al. [2007, 2009]. Tsai et al. [2009] is the journal version of Tsai et al. [2007], which contained more complete and detailed information, so *Tsai et al. [2009]* is selected as the primary study.

Table XVI. Data Extraction Form

ID	Field	Description	Research Question
Content Information of Article			
1	Title	Title of the primary study	
2	Abstract	Abstract of the primary study	GQ2
3	Objectives	What are the objectives of the study?	GQ2
4	Test Executer	Who performs the test execution?	GQ1
5	SUT	What is the system to be tested?	GQ3
6	Standards	Which standards (or techniques) are being researched?	GQ3
7	Testing Method	Which regression testing techniques are applied to SUT?	GQ4[a]
8	Validation	Which is the method used to validate the study?	GQ5
9	Services	Services used for empirical study	GQ5
10	Validity	Limitation, threat to validity	
11	Conclusion	Conclusion of the study	GQ2
Reference Information of Article			
13	Authors	Authors of the primary study	
14	Year	Publication year of the primary study	SQ1
15	Type	Publication type of the primary study	SQ2
16	Source	Name of publication where the primary study was published	SQ2
Extraction Information of Article			
17	Reference	Unique reference for the primary study	
18	Date	Date of the data extraction	

[a]The answers for FQ1, FQ2, and FQ3 can be partially extracted from GQ4.

—Zhai et al. [2014], Zhai and Chan [2010], and Zhai et al. [2010]. Zhai et al. [2014] is the journal version of Zhai and Chan [2010] and Zhai et al. [2010], so *Zhai et al. [2014]* is selected as the primary study.
—Ruth and Tu [2007c, 2007b], Ruth et al. [2007], and Ruth and Tu [2008]. Ruth and Tu [2007c] is the poster version of *Ruth et al. [2007]*, so we keep Ruth et al. [2007] first. In addition, Ruth and Tu [2008] contains the content of Ruth and Tu [2007b] and it focused on the empirical study on the approach proposed by Ruth et al. [2007]. Hence, we combine *Ruth et al. [2007]* and *Ruth and Tu [2008]* together as a single primary study.
—Ruth [2008] and Ruth and Tu [2007a]. Ruth and Tu [2007a] is the poster version of Ruth [2008], so *Ruth [2008]* is selected as the primary study.
—Khan and Heckel [2009] and Khan and Heckel [2011]. Khan and Heckel [2011] is the extended version of Khan and Heckel [2009], so *Khan and Heckel [2011]* is selected as the primary study.

C. QUALITY ASSESSMENT RESULTS

The results of quality assessment are listed in Table XV.

D. DATA EXTRACTION FORM

The data extraction form is shown in Table XVI.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for useful feedback on an earlier version of this article.

REFERENCES

W. Richards Adrion, Martha A. Branstad, and John C. Cherniavsky. 1982. Validation, verification, and testing of computer software. *ACM Comput. Surv.* 14, 2 (1982), 159–192.

Wasif Afzal, Richard Torkar, and Robert Feldt. 2009. A systematic review of search-based testing for non-functional system properties. *Inf. Softw. Technol.* 51, 6 (2009), 957–976.

A. Askarunisa, A. M. Abirami, K. Arockia Jackulin Punitha, B. Karthik Selvakumar, and R. Arunkumar. 2010. Sequence-based techniques for black-box test case prioritization for composite service testing. In *Proceedings of the 2010 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC'10)*. 1–4.

A. Askarunisa, K. A. J. Punitha, and A. M. Abirami. 2011. Black box test case prioritization techniques for semantic based composite web services using OWL-S. In *Proceedings of the 2011 International Conference on Recent Trends in Information Technology (ICRTIT'11)*. 1215–1220.

B. Athira and P. Samuel. 2010. Web services regression test case prioritization. In *Proceedings of the 2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM'10)*. 438–443.

Xiaoying Bai and Ron S. Kenett. 2009. Risk-Based adaptive group testing of semantic web services. In *Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09)*. 485–490.

Cesare Bartolini, Antonia Bertolino, Eda Marchetti, and Andrea Polini. 2008. Towards automated WSDL-Based testing of web services. In *Proceedings of the 6th International Conference on Service-Oriented Computing (ICSOC'08)*. 524–529.

P. Benedusi, A. Cmitile, and U. De Carlini. 1988. Post-Maintenance testing based on path change analysis. In *Proceedings of the International Conference on Software Maintenance (ICSM'88)*. 352–361.

Mustafa Bozkurt. 2013. Cost-aware pareto optimal test suite minimisation for service-centric systems. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO'13)*. 1429–1436.

Mustafa Bozkurt, Mark Harman, and Youssef Hassoun. 2013. Testing and verification in service-oriented architecture: a survey. *Software Testing, Verification and Reliability* 23, 4 (2013), 261–313.

Gerardo Canfora and Massimiliano Di Penta. 2006. Testing services and service-centric systems: Challenges and opportunities. *IT Professional* 8, 2 (2006), 10–17.

Gerardo Canfora and Massimiliano Di Penta. 2009. Service-Oriented architectures testing: A survey. In *Software Engineering*. Springer, 78–105.

Lin Chen, Ziyuan Wang, Lei Xu, Hongmin Lu, and Baowen Xu. 2010. Test case prioritization for web service regression testing. In *Proceedings of the 2010 5th IEEE International Symposium on Service Oriented System Engineering (SOSE'10)*. 173–178.

Subhankar Dhar and Upkar Varshney. 2011. Challenges and business models for mobile location-based services and advertising. *Commun. ACM* 54, 5 (2011), 121–128.

Massimiliano Di Penta, Marcello Bruno, Gianpiero Esposito, Valentina Mazza, and Gerardo Canfora. 2007. Web services regression testing. In *Test and Analysis of Web Services*. Springer, 205–234.

Wenli Dong. 2008. Test case reduction technique for BPEL-based testing. In *Proceedings of the 2008 International Symposium on Electronic Commerce and Security (ISECS'08)*. 814–817.

Sebastian Elbaum, Alexey G. Malishevsky, and Gregg Rothermel. 2002. Test case prioritization: A family of empirical studies. *IEEE Trans. Softw. Eng.* 28, 2 (2002), 159–182.

Emelie Engström, Per Runeson, and Mats Skoglund. 2010. A systematic review on regression test selection techniques. *Inf. Softw. Technol.* 52, 1 (2010), 14–30.

Todd L. Graves, Mary Jean Harrold, Jung Min Kim, Adam Porter, and Gregg Rothermel. 2001. An empirical study of regression test selection techniques. *ACM Trans. Softw. Eng. Methodol.* 10, 2 (2001), 184–208.

Mary Jean Harrold, James A. Jones, Tongyu Li, Donglin Liang, Alessandro Orso, Maikel Pennings, Saurabh Sinha, S. Alexander Spoon, and Ashish Gujarathi. 2001. Regression test selection for Java software. In *Proceedings of the 16th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'01)*. 312–326.

Shan Shan Hou, Lu Zhang, Tao Xie, and Jia Su Sun. 2008. Quota-Constrained test-case prioritization for regression testing of service-centric systems. In *Proceedings of the 24th IEEE International Conference on Software Maintenance (ICSM'08)*. 257–266.

R. Kavitha and Dr. N. Sureshkumar. 2010. Test case prioritization for regression testing based on severity of fault. *Int. J. Comput. Sci. Engin.g* 2, 5 (2010), 1462–1466.

Tamim Ahmed Khan and Reiko Heckel. 2009. A methodology for model-based regression testing of web services. In *Proceedings of the 2009 Testing: Academic and Industrial Conference - Practice and Research Techniques (TAIC-PART'09)*. 123–124.

Tamim Ahmed Khan and Reiko Heckel. 2011. On model-based regression testing of web-services using dependency analysis of visual contracts. In *Proceedings of the 14th International Conference on Fundamental Approaches to Software Engineering: Part of the Joint European Conferences on Theory and Practice of Software (FASE'11/ETAPS'11)*. 341–355.

Barbara A. Kitchenham and S. Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Technical Report EBSE-2007-01. Keele University.

Bixin Li, Dong Qiu, Shunhui Ji, and Di Wang. 2010. Automatic test case selection and generation for regression testing of composite service based on extensible BPEL flow graph. In *Proceedings of the 26th IEEE International Conference on Software Maintenance (ICSM'10)*. 1–10.

Bixin Li, Dong Qiu, Hareton Leung, and Di Wang. 2012. Automatic test case selection for regression testing of composite service based on extensible Bpel flow graph. *J. Syst. Softw.* 85, 6 (2012), 1300–1324.

Zheng Li, Mark Harman, and Robert M. Hierons. 2007. Search algorithms for regression test case prioritization. *IEEE Trans. Softw. Eng.* 33, 4 (2007), 225–237.

Z. J. Li, H. F. Tan, H. H. Liu, J. Zhu, and N. M. Mitsumori. 2008. Business-Process-Driven gray-box SOA testing. *IBM Syst. J.* 47, 3 (2008), 457–472.

Hehui Liu, Zhongjie Li, Jun Zhu, and Huafang Tan. 2007. Business process regression testing. In *Proceedings of the 5th International Conference on Service-Oriented Computing (ICSOC'07)*. 157–168.

Lijun Mei, Yan Cai, Changjiang Jia, Bo Jiang, and W. K. Chan. 2013a. Prioritizing structurally complex test pairs for validating WS-BPEL evolutions. In *Proceedings of the 20th International Conference on Web Services (ICWS'13)*. 147–154.

Lijun Mei, Yan Cai, Changjiang Jia, Bo Jiang, and W. K. Chan. 2013b. Test pair selection for test case prioritization in regression testing for WS-BPEL programs. *Int. J. Web Serv. Res.* 10, 1 (2013), 73–102.

Lijun Mei, W. K. Chan, and T. H. Tse. 2008. An adaptive service selection approach to service composition. In *Proceedings of the IEEE International Conference on Web Services (ICWS'08)*. 70–77.

Lijun Mei, W. K. Chan, T. H. Tse, and Robert G. Merkel. 2009. Tag-Based techniques for black-box test case prioritization for service testing. In *Proceedings of the 2009 9th International Conference on Quality Software (QSIC'09)*. 21–30.

Lijun Mei, W. K. Chan, T. H. Tse, and Robert G. Merkel. 2011. XML-Manipulating test case prioritization for XML-manipulating services. *J. Syst. Softw.* 84, 4 (2011), 603–619.

Lijun Mei, Ke Zhai, Bo Jiang, W. K. Chan, and T. H. Tse. 2012. Preemptive regression test scheduling strategies: A new testing approach to thriving on the volatile service environments. In *Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC'12)*. 72–81.

Lijun Mei, Zhenyu Zhang, W. K. Chan, and T. H. Tse. 2009. Test case prioritization for regression testing of service-oriented business applications. In *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*. 901–910.

Cu D. Nguyen, Alessandro Marchetto, and Paolo Tonella. 2011a. Change sensitivity based prioritization for audit testing of webservice compositions. In *Proceedings of the 2011 IEEE 4th International Conference on Software Testing, Verification and Validation Workshops (ICSTW'11)*. 357–365.

Cu D. Nguyen, Alessandro Marchetto, and Paolo Tonella. 2011b. Test case prioritization for audit testing of evolving web services using information retrieval techniques. In *Proceedings of the 2011 IEEE International Conference on Web Services (ICWS'11)*. 636–643.

Akira K. Onoma, Wei-Tek Tsai, Mustafa Poonawala, and Hiroshi Suganuma. 1998. Regression testing in an industrial environment. *Commun. ACM* 41, 5 (1998), 81–86.

Marcos Palacios, José García-Fanjul, and Javier Tuya. 2011. Testing in service oriented architectures with dynamic binding: A mapping study. *Inf. Softw. Technol.* 53, 3 (2011), 171–189.

Mark Petticrew and Helen Roberts. 2005. *Systematic Reviews in the Social Sciences: A Practical Guide*. Wiley.

Gregg Rothermel and Mary Jean Harrold. 1996. Analyzing regression test selection techniques. *IEEE Trans. Softw. Eng.* 22, 8 (1996), 529–551.

Gregg Rothermel and Mary Jean Harrold. 1997. A safe, efficient regression test selection technique. *ACM Trans. Softw. Eng. Methodol.* 6, 2 (1997), 173–210.

Gregg Rothermel, Mary Jean Harrold, Jeffery von Ronne, and Christie Hong. 2002. Empirical studies of test-suite reduction. *Softw. Test. Verif. Reliab.* 12, 4 (2002), 219–249.

Gregg Rothermel, Roland J. Untch, and Chengyun Chu. 2001. Prioritizing test cases for regression testing. *IEEE Trans. Softw. Eng.* 27, 10 (2001), 929–948.

Michael Ruth and Shengru Tu. 2007a. Concurrency issues in automating RTS for web services. In *Proceedings of the 2007 IEEE International Conference on Web Services (ICWS'07)*. 1142–1143.

Michael Ruth and Shengru Tu. 2007b. A safe regression test selection technique for web services. In *Proceedings of the 2nd International Conference on Internet and Web Applications and Services (ICIW'07)*. 47–52.

Michael E. Ruth. 2008. Concurrency in a decentralized automatic regression test selection framework for web services. In *Proceedings of the 15th ACM Mardi Gras Conference (MG'08)*. 7:1–7:8.

Michael E. Ruth. 2011. Employing privacy-preserving techniques to protect control-flow graphs in a decentralized, end-to-end regression test selection framework for web services. In *Proceedings of the 2011 IEEE 4th International Conference on Software Testing, Verification and Validation Workshops (ICSTW'11)*. 139–148.

Michael E. Ruth, Feng Lin, and Shengru Tu. 2006. Applying safe regression test selection techniques to java web services. *Int. J. Web Service Pract.* 2, 1–2 (2006), 1–10.

Michael E. Ruth, Sehun Oh, Adam Loup, Brian Horton, Olin Gallet, Marcel Mata, and Shengru Tu. 2007. Towards automatic regression test selection for web services. In *Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC'07)*. 729–736.

Michael E. Ruth and Curtis Rayford. 2011. A privacy-aware, end-to-end, CFG-Based regression test selection framework for web services using only local information. In *Proceedings of the 2011 4th International Conference on the Applications of Digital Information and Web Technologies (ICADIWT'11)*. 13–18.

Michael E. Ruth and Shengru Tu. 2007c. Towards automating regression test selection for web services. In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)*. 1265–1266.

Michael E. Ruth and Shengru Tu. 2008. Empirical studies of a decentralized regression test selection framework for web services. In *Proceedings of the 2008 Workshop on Testing, Analysis, and Verification of Web Services and Applications (TAV-WEB'08)*. 8–14.

Mary Shaw. 2003. Writing good software engineering research papers: Minitutorial. In *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*. 726–736.

A. Tarhini, H. Fouchal, and N. Mansour. 2006. Regression testing web services-based applications. In *Proceedings of the 2006 IEEE International Conference on Computer Systems and Applications (AICCSA'06)*. 163–170.

Wei Tek Tsai, Yinong Chen, Raymond Paul, Hai Huang, Xinyu Zhou, and Xiao Wei. 2005. Adaptive testing, oracle generation, and test case ranking for web services. In *Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05)*. 101–106.

W. T. Tsai, Y. Chen, R. Paul, N. Liao, and H. Huang. 2004. Cooperative and group testing in verification of dynamic composite web services. In *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*. 170–173.

Wei Tek Tsai, Xinyu Zhou, Raymond A. Paul, Yinong Chen, and Xiaoying Bai. 2009. A coverage relationship model for test case selection and ranking for multi-version software. In *High Assurance Services Computing*. Springer, 285–311.

W. T. Tsai, Xinyu Zhou, Raymond A. Paul, Yinong Chen, and Xiaoying Bai. 2007. A coverage relationship model for test case selection and ranking for multi-version software. In *Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*. 105–112.

Di Wang, Bixin Li, and Ju Cai. 2008. Regression testing of composite service: An XBFG-based approach. In *Proceedings of the 2008 IEEE Congress on Services (SERVICES'08)*. 112–119.

Chunyang Ye, S. C. Cheung, W. K. Chan, and Chang Xu. 2009. Atomicity analysis of service composition across organizations. *IEEE Trans. Softw. Eng.* 35, 1 (2009), 2–28.

S. Yoo and M. Harman. 2012. Regression testing minimization, selection and prioritization: A survey. *Softw. Test. Verif. Reliab.* 22, 2 (2012), 67–120.

Zulfa Zakaria, Rodziah Atan, Abdul Azim Abdul Ghani, and Nor Fazlida Mohd. Sani. 2009. Unit testing approaches for BPEL: A systematic review. In *Proceedings of the 2009 16th Asia-Pacific Software Engineering Conference (APSEC'09)*. 316–322.

Ke Zhai and W. K. Chan. 2010. Point-of-Interest aware test case prioritization: Methods and experiments. In *Proceedings of the 2010 10th International Conference on Quality Software (QSIC'10)*. 449–456.

Ke Zhai, Bo Jiang, and WK Chan. 2012. Prioritizing test cases for regression testing of location-based services: Metrics, techniques and case study. *IEEE Trans. Serv. Comput.* 7, 1, 54–67.

Ke Zhai, Bo Jiang, W. K. Chan, and T. H. Tse. 2010. Taking advantage of service selection: A study on the testing of location-based web services through test case prioritization. In *Proceedings of the 2010 IEEE International Conference on Web Services (ICWS'10)*. 211–218.

Lingming Zhang, Dan Hao, Lu Zhang, Gregg Rothermel, and Hong Mei. 2013. Bridging the gap between the total and additional test-case prioritization strategies. In *Proceedings of the 35th International Conference on Software Engineering (ICSE'13)*. 192–201.

Received May 2013; revised April 2014; accepted May 2014