UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by **Novática** <http://www.ati.es/novatica/>, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática,* <http://www.ati.es/>)

UPGRADE monographs are also published in Spanish (full version printed; summary, abstracts and some articles online) by **Novática**

UPGRADE was created in October 2000 by CEPIS and was first published by **Novática** and **INFORMATIK/INFORMATIQUE**, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>)

UPGRADE is the anchor point for UPENET (UPGRADE European NETwork), the network of CEPIS member societies' publications, that currently includes the following ones:
- **Informatica**, journal from the Slovenian CEPIS society SDI
- **Informatik-Spektrum**, journal published by Springer Verlag on behalf of the CEPIS societies GI, Germany, and SI, Switzerland
- ITNOW, magazine published by Oxford University Press on behalf of the British CEPIS society BCS
- **Mondo Digitale**, digital journal from the Italian CEPIS society AICA
- **Novática**, journal from the Spanish CEPIS society ATI
- **OCG Journal**, journal from the Austrian CEPIS society OCG
- **Pliroforiki**, journal from the Cyprus CEPIS society CCS
- **Tölvumál**, journal from the Icelandic CEPIS society ISIP

Cover page designed by Concha Arias-Pérez
"Full Steam Ahead" / © CEPIS 2009
Layout Design: François Louis Nicolet
Composition: Jorge Llácer-Gil de Ramales

Editorial correspondence: Llorenç Pagés-Casas <pages@ati.es>
Advertising correspondence: <novatica@ati.es>

UPGRADE Newslist available at
<http://www.upgrade-cepis.org/pages/editinfo.html#newslist>

ISSN 1684-5285

Monograph of next issue (December 2009)

"Privacy and Identity Management"

(The full schedule of **UP**GRADE is available at our website)

CEPIS UPGRADE

The European Journal for the Informatics Professional
http://www.upgrade-cepis.org

Vol. X, issue No. 5, October 2009

Monograph: Experiences and Advances in Software Quality
(published jointly with Novática*)
Guest Editors: *Darren Dalcher and Luis Fernández-Sanz*

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by **Novática**, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*) at <http://www.ati.es/novatica/>.

Methods for Testing Web Service Compositions

José García-Fanjul, Marcos Palacios-Gutiérrez, Javier Tuya-González, and Claudio de la Riva-Alvarez

The deployment of software as a service has the objective, in the short or medium term, that these services will be invoked not just from one particular application, but also from other software or services. Consequently, using well-established and automated testing methods is essential to firstly assure the quality of the deployed services and also to facilitate regression testing. In this paper, we describe methods that have been recently proposed to test web service compositions, particularly focusing on the de-facto industrial standard BPEL.

Keywords: BPEL**,** Dynamic Binding, Monitoring, Software Testing, Web Service Compositions.

Authors

José García-Fanjul received the PhD degree in Computing from the Computer Science Department of the University of Oviedo, in Spain, where he is, currently, an associate professor. His research interests focus on the field of Software Engineering and, more specifically, software testing for Service Oriented Architectures. <jgfanjul@uniovi.es>.

Marcos Palacios-Gutiérrez received the MS degree in Computer Science from the University of Oviedo in 2008 where he is, currently, a PhD candidate in computer science. His research interests are in the field of software engineering, especially in software testing and service-oriented architectures. <palacios@lsi.uniovi.es>.

Javier Tuya-González received the PhD degree from the University of Oviedo, in Spain, where he is, currently, an associate professor in the Computer Science Department. His research interests focus on the field of Software Engineering and, more specifically, on Software Testing for Service Oriented Architectures and database applications. He has published several articles in international conferences and journals. He is a member of a number of professional associations such as IEEE, IEEE/CS, ACM, Association for Software Testing and SISTEDES. Currently, he is a member of the ISO Workgroup JTC1/SC7/WG26 - Software Testing, which elaborates the new ISO/IEC 29119 standard. He also serves as coordinator of the corresponding Spanish group AEN/CTN71/SC7/GT 26. <tuya@uniovi.es>.

Claudio de la Riva-Álvarez received the PhD degree in Computing from the Computer Science Department of the University of Oviedo, in Spain, where he is, currently, an associate professor. His research interests include Software Testing, Verification and Validation. <claudio@uniovi.es>.

1 Introduction

Service Oriented Architectures (SOAs) have become, in recent years, a common standard for building and deploying software. In such architectures, software is divided into decoupled services which are composed to build complex applications. As with any new technology, SOAs have a set of advantages such as the low coupling between the services. This characteristic enables binding with third-party provided services or even establish dynamic binding policies. However, the adoption of this new technology has also raised concerns regarding the software development processes and, more precisely, software testing processes. Canfora and Di Penta [1] and Zhang and Zhang [2] identified a number of unresolved challenges in the application of traditional software testing technologies to services such as:

1. The need to remotely test web services, with its associated cost.
2. The impact that the limited information exposed about a web service has on the design of test cases.
3. The ability to dynamically search and invoke web services.

Since these challenges, among others, were identified, research groups have worked on the definition of testing technologies useful for Service Oriented Architectures.

This is a state of the art article on the testing methods committed specifically to uncover faults in web service compositions, particularly focusing on the de-facto industrial standard BPEL.

The article is organized as follows: Section 2 outlines the background concepts regarding services, compositions of services and software testing. The following three sections are dedicated to monitoring methods (Section 3), methods to test dynamic binding compositions (Section 4) and functional testing (Section 5). A discussion is given in Section 6 regarding common issues in research about software testing for SOAs and lastly, in Section 7, conclusions are presented.

2 Background

2.1 Service Oriented Architectures

A good definition of web service is given in the web services glossary made public by the W3C consortium [3]:

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

As it is said in the definition, most of the existing services today are web services and thus they are invoked using the HTTP (*Hypertext Transfer Protocol*) protocol. Furthermore, most of the services today have their interfaces described in terms of the WSDL (*Web Services Description Language*) standard and they exchange data in XML (*eXtensible Markup Language*) format. Nevertheless, and bearing in mind that there exists a great amount of deployed services which are not web services (as they can be invoked, for example, using a protocol other than HTTP) in this article we tend to use the term "service" and not the more restrictive "web service".

Service providers may publish their characteristics in brokers or registries that commonly use the UDDI (*Universal Description, Discovery and Integration*) standard. Such brokers may be queried by applications that need to use a certain functionality, and they will reply with a number of suitable services. Applications will then select a service, dynamically bind to it and invoke it. This software design is called Service Oriented Architecture (SOA).

A composition of services is distributed software that invokes an amount of services such that services may come from different providers. A composition of services can be executing different software each time it is executed, because it cannot be controlled if providers change the implementation of their services. Furthermore, in dynamic binding contexts, a composition of services may be executing different services each time it is executed.

Compositions of services are classified conceptually, as orchestrations or choreographies. Orchestrations require that there is one service which invokes all the other services in the composition. The de facto standard to implement orchestrations of web services is BPEL (*Business Process Execution Language*) [4]. Service choreographies do not need a central coordinator, and the coordination responsibility is shared among all the participating services.

2.2 Software Testing

Testing is one of the fundamental processes for quality control of software. One of the best definitions for software testing comes from the IEEE standard Std610.12-1990 [5]:

The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software items.

It is commonly understood that testing implies that the software under test should be executed (dynamic testing). However, the above definition includes any analysis made to a software item as a test, and therefore it also includes static checking techniques, in which software is not executed to find bugs.

The traditional approach to software testing required that most of the tests should be executed before the software is deployed in production environments. The point was that quality control processes should be performed before the product is released to customers. Nowadays, due to the emergence of Service Oriented Architectures, this approach is complemented with monitoring techniques - also called online testing [6]. These techniques are oriented towards continuous inspection of software behaviour in production environments, and are often complemented with self-adaptation strategies, in case misbehaviour is detected.

3 Monitoring

One of the main characteristics of Service Oriented Architectures is low coupling. Service providers may introduce changes in the behaviour of their services in a number of ways such as, for example, deploying a new version of the service, or modifying computational resources dedicated to the execution of services.

Thus, from the point of view of service clients, they must define strategies to find out, in runtime, whether there are misbehaviours in their service compositions due to changes in the behaviour of the composed services. As mentioned above, the testing techniques oriented towards continuous monitoring of software behaviour in production environments are monitoring techniques.

As with any process that must be executed in production settings, monitoring techniques should be able to detect faults by observing the execution of the composition of services with a minimum interference in the behaviour or workload of the composition. Monitoring techniques that can be implemented with no change in the behaviour of the composition or the services which are under test are often called passive monitoring.

In this kind of technique, it is common to deploy probes that inspect the messages (typically SOAP, - *Simple Object Access Protocol* -) that services participating in the composition interchange. A fault is therefore detected when the obtained messages do not comply with the specification of the service composition. For example, Raimondi et al. [7] propose deploying probes in service providers or clients. The probes check whether Service Level Agreements (SLAs) are met in terms of quality of service (QoS) properties. Their paper describes how to systematically obtain probes from SLAs specified in the SLAng language.

A similar proposal [8] describes an architecture of probes that monitor functional and non-functional behaviours in service compositions. The difference between this and the former proposal is that the specification of the expected behaviour for the composition of services is made in an augmented version of Finite State Machines.

In some situations, passive monitoring techniques cannot be applied. Therefore, the composition or the services must be modified to instrument their code and obtain information about their behaviour when executed. These techniques are called active monitoring. A relevant proposal in active monitoring [9] describes how to deploy a tool to monitor the execution of BPEL processes. The tester defines rules that specify the expected behaviour of the composition, and these rules are inserted into the BPEL code as comments. The BPEL engine is complemented with a plug-in that is able to interpret such rules and decide whether the observed behaviour conforms to the rules or not.

The papers described above use the probes to discover faults as soon as the faulty software is executed. This is the most common approach for monitoring compositions of services, but some faults cannot be discovered using such approaches. Offline monitoring techniques gather information about the behaviour of compositions of services and then offline process that data to establish whether there are faults.

The most representative example is the article by Rozinat and Van der Aalst [10]. In this article, they propose a method to determine, using Petri Net based tools, whether the executions of a composition of services conform to a BPEL specification of the composition.

4 Testing Dynamic Binding Compositions

A key feature in SOA architectures is their capability to change the services that participate in a composition. Service clients may decide, in runtime, which service will be invoked among the suitable ones. This characteristic is called "dynamic binding" and it is implemented using brokers (or registries) of services. Service providers publish the features of their services in the broker, and service clients query the broker and decide which of the available services will be invoked. Figure 1 illustrates this process.

Dynamic binding raises certain challenges from the point of view of software testing. For example, brokers may wish to decide whether the services they publish hold certain QoS properties. Test suites may then be designed and successfully executed on candidate services as a precondition to be referenced by the broker [12]. The broker may also periodically monitor to ensure that services are continuously complying with the defined requirements in terms of QoS [13].

Both of the above mentioned papers propose to extend the UDDI standard, as UDDI defines the functionality of the broker as a passive registry. UDDI brokers just store the necessary information to realize the dynamic binding, and not semantic data related to the behaviour of the referenced services.

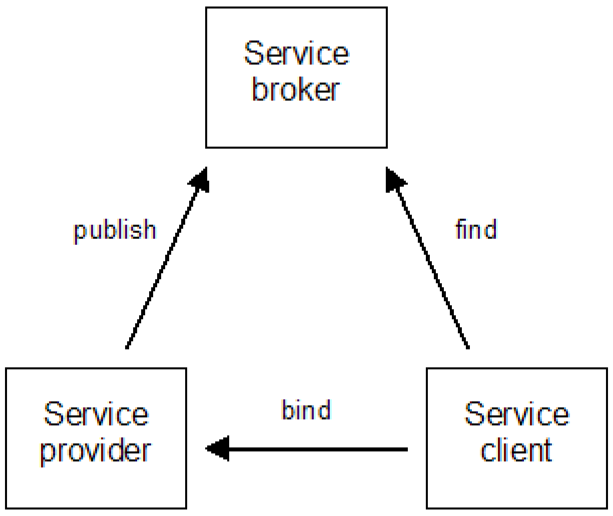

Figure 1: Service Brokering - adapted from Papazoglou et al. [11].

Other proposals take advantage of the dynamic nature of service invocations so that when a fault is detected in a monitored composition of web services, the invoked service may be changed for another one that successfully executes a test suite, and so it is assumed that it complies with the specification. For example, Moser et al. [14] have designed a tool to replace, in runtime, services which are composed in a BPEL process. The replacement is performed when the monitoring framework identifies that QoS criteria do not hold. In another approach a framework is described to decide which of the available services will be selected to be bound in a composition, using genetic algorithms to estimate the best solution in terms of QoS [15].

5 Functional Testing

There is an immense diversity of situations that may be reproduced as a test with the intention of uncovering faults in software functionality. In practice and in most of the projects, the amount of different test situations may be considered infinite. Thus, functional test designers must select test cases that, when executed, have a high probability of finding faults and, also, can be fully designed and executed within the available resources. Research in functional testing and test case generation has, therefore, the objective of finding good test suites for testing a given software, that optimize the above described criteria.

Commonly, approaches to generate functional test cases for software rely on techniques that make a specification of the expected behaviour of the software under test, and test cases are derived from the specification applying certain algorithms. Specifically, in the field of functional testing for compositions of services, model checking (a formal method) has been used in several approaches [15][16]. In the first of these papers, a model is obtained from the specification of the composition in BPEL, and an adequacy criterion (transition coverage) is defined to systematically obtain test cases applying a model checker. In the latter, the composition is specified in OWL-S and no criteria are defined to obtain the test cases: they are obtained from properties which are specified by hand.

Other approaches use different formalisms to obtain the test cases. Mei et al. [17] describes a data-flow technique to obtain test cases for BPEL processes relying on term-rewriting tools. Another paper prescribes a control-flow method and expounds how to generate tests, also for BPEL processes, from a model of the flow of BPEL activities and using a constraint solver [18]. Another technique, Petri Nets, that is commonly used to generate test cases for software testing, is applied to test compositions of web services in which the desired behaviour is extracted from contracts [19].

The majority of the research conducted on functional testing for SOAs is dedicated to the unit testing of a service or, at most, a service that is executed in the scope of a composition of services. To isolate the service under test from the rest of the composition, other services must be deployed to work as stubs. The design and implementation of such stub services may be automatically feasible if the behaviour of the composition is specified [20]. Other frameworks and tools for designing and executing unit tests for services are described in the papers of Li et al. [21] and Mayer and Lübke [22]. In both it is possible to unit test services that participate in a composition that is specified using BPEL. The work of Mayer and Lübke was later implemented into a tool called BPELUnit [23].

6 Discussion

For Service Oriented Architectures, there exist standards that are commonly accepted to describe the interfaces of the services. The growing enthusiasm for these standards in industrial settings is one of the key advantages of these architectures. But up to now no consensus has been reached with respect to standards to describe service behaviour, which is not a new problem [24].

If the papers that are referenced in this article are reviewed, most of them use one technique or another to specify behaviour of services or compositions of services, such as BPEL, OWL-S or SLAng just to mention three. Some of the researchers even propose new ad-hoc techniques to specify behaviour. In the field of testing techniques, the lack of commonly accepted standards to specify service behaviour leads to difficulties in establishing systematic strategies for the execution of tests, adequacy criteria for selecting test cases or the definition of oracles to set the expected behaviour of test cases. One of the fields in which the lack of these standards is more visible is in the testing of dynamic binding compositions of services. The de facto standard for broker architecture (UDDI) is systematically complemented in research papers, with semantic information that allows registering or querying services applying different criteria depending on the service behaviour.

New standards must be adopted to define functional and non functional behaviour of services and compositions of services. Nevertheless, it must be noted that the latter (non functional) characteristics are the ones receiving more attention in research works in the fields of monitoring and dynamic binding.

This trend may be influenced by two concerns that final users commonly state about SOAs. First of all, the need to show that SOA solutions, like any other software, fulfill a number of QoS requirements to warrant that it is usable, before considering whether there are problems in the functional requirements. On the other hand, several final users feel that functional faults are easier to detect and correct than QoS faults, so more resources are employed in availability or performance testing rather than in functional testing.

It should also be stressed that many of the papers published in the field of testing SOAs are not validated against industrial-size examples. Many of them are just exemplified with simple compositions of services that are extracted from standards or designed ad-hoc for the research. It is of the utmost importance to define standard examples that can be used to validate the researches, and compare the results obtained on the application of different testing techniques. Industrial sized examples must also be published to evaluate the scalability of the research approaches.

7 Conclusions

In recent years, researchers have worked on different techniques to test compositions of services. One of the fields in which there have been a greater number of proposals is the monitoring of SOAs, an especially relevant matter, bearing in mind the low-coupled nature of services.

Regarding the testing of dynamic binding compositions, there are just a few research proposals and much work needs to be done in the future, even at the conceptual level. A great amount of work has been published about functional testing of SOAs, using techniques that were productive in the past for other kinds of software, but there is also a need to focus on the quirks of compositions of services.

Advances in the testing of SOA software could be made if there were commonly adopted standards for the specification of the behaviour of services. The only standard that has been clearly adopted at the industrial level has been BPEL. This language allows specification of the functional behaviour of compositions of web services but there is a growing need to define and adopt standards for the definition of functional and non functional properties for individual services.

Acknowledgements

This work is supported by the Ministry of Science and Innovation (Spain) under the National Program for Research, Development and Innovation, projects Test4SOA (TIN2007-67843-C06-01) and RePRIS (TIN2007-30391-E).

References

[1] G. Canfora, M. Di Penta. Testing services and service-centric systems: challenges and opportunities. IT Professional 2006; 8(2):10–17.

[2] J. Zhang, L.J. Zhang. Web services quality testing. Int. Journal of Web Services Research 2005; 2(2):1-4.

[3] W3C. Web Services Glossary. <http://www.w3.org/TR/ws-gloss/>.

[4] IBM. Business Process Execution Language for Web Services version 1.1. <http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>.

[5] IEEE. IEEE Std 610.12-1990, IEEE standard glossary of software engineering terminology. <http://standards.ieee.org>.

[6] A. Bertolino. Software testing research: achievements, challenges, dreams. Proceedings of the Workshop on the Future of Software Engineering (FOSE), 2007; Minneapolis (USA); pp. 85-103.

[7] F. Raimondi, J. Skene, W. Emmerich. Efficient online monitoring of web-service SLAs. Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), 2008;

Atlanta (Georgia – USA); pp. 170-180.

[8] A. Benharref, R. Dssouli, M. Adel Serhani, A. En-Nouaary, R.H. Glitho. Efficient traces' collection mechanisms for passive testing of web services. Information and Software Technology 2009; 51:362-374.

[9] L. Baresi, S. Guinea. Towards dynamic monitoring of WS-BPEL processes. Proceedings of the Third International Conference on Service-Oriented Computing (ICSOC), 2005; Amsterdam (Holanda); pp. 269-282.

[10] A. Rozinat, W.M.P. Van der Aalst. Conformance checking of processes based on monitoring real behavior. Information Systems 2008; 33(1):64-95.

[11] M.P. Papazoglou, W.J. Van den Heuvel. Service oriented architectures: approaches, technologies and research issues. The VLDB Journal 2007; 16:389-415.

[12] A. Bertolino, A. Polini. The Audition Framework for testing web services interoperability. Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 2005; Porto (Portugal); pp. 134-142.

[13] X. Bai, S. Lee, W.T. Tsai, Y. Chen. Collaborative web services monitoring with active service broker. Proceedings of the IEEE International Computer Software and Applications Conference (COMPSAC), 2008; Turku (Finlandia); pp. 84-91.

[14] O. Moser, F. Rosenberg, S. Dustdar. Non-intrusive monitoring and service adaptation for WS-BPEL. Proceedings of the Int. World Wide Web Conference (WWW), 2008; Beijing (China); pp. 815-824.

[15] J. García-Fanjul, J. Tuya, C. De la Riva. Generating test cases specifications for BPEL compositions of web services using SPIN. Proceedings of the Int. Workshop on Web Services - Modeling and Testing, 2006; Palermo (Italia); pp. 83-94.

[16] H. Huang, W.T. Tsai, R. Paul, Y. Chen. Automated model checking and testing for composite web services. Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2005; Seattle (USA); pp. 300-307.

[17] L. Mei, W.K. Chan, T.H. Tse. Data flow testing of service-oriented workflow applications. Proceedings of the 30th International Conference on Software Engineering (ICSE), 2008; Leipzig (Alemania); pp. 371-380.

[18] J. Yan, Z. Li, Y. Yuan, W. Sun, J. Zhang. BPEL4WS unit testing: test case generation using a concurrent path analysis approach. Proceedings of the 17th International Symposium on Software Reliability Engineering (ISSRE), 2006; Raleigh (North Carolina - USA); pp. 75-84.

[19] G. Dai, X. Bai, Y. Wang, F. Dai. Contract-based testing for web services. Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC), 2007; Beijing (China); pp. 517-526.

[20] A. Bertolino, G. De Angelis, L. Frantzen, A. Polini. Model-based generation of testbeds for web services. Proceedings of the 20th International Conference on Testing of Software and Communicating Systems and 8th International Workshop on Formal Approaches to Testing of Software (TestCom/FATES), 2008; Tokyo (Japón); pp. 266-282.

[21] Z. Li, W. Sun, Z.B. Jiang, X. Zhang. BPEL4WS unit testing: framework and implementation. Proceedings of the IEEE International Conference on Web Services (ICWS), 2005; Orlando (USA); pp. 103-110.

[22] P. Mayer, D. Lübke. Towards a BPEL unit testing framework. Proceedings of the Workshop on Testing, Analysis, and Verification of Web Services and Applications (TAV-WEB), 2006; Portland (Maine – USA); pp. 33-42.

[23] Leibniz Universität Hannover. BPELUnit - The Open Source Unit Testing Framework for BPEL. <http://www.se.uni-hannover.de/forschung/soa/bpelunit/>.

[24] S. Jones. Toward an acceptable definition of service. IEEE Software 2005; 22(3):87-93.