Framework for Dynamic Web Services Composition Guided by Live Testing

Mounia Elqortobi(✉), Jamal Bentahar, and Rachida Dssouli

CIISE, Concordia University, 1455 Boulevard de Maisonneuve O,
Montreal, QC H3G 1M8, Canada
m_elqort@live.concordia.ca,
{jamal.bentahar,rachida.dssouli}@concordia.ca

Abstract. Web services allow businesses to offer their services and consumers to retrieve and use them. Businesses own some services and can reuse services that belong to other businesses to perform new transactional activities. By doing this, they achieve outsourcing, cost, and resources optimization. The advances in design principles, architectures, protocols and languages have helped to solve some of the problems related to the composition of business applications. Web service composition technology emerged as a new approach for efficient automation and integration of business processes based on Service-Oriented Architecture (SOA). SOA provides a set of principles to create distributed computing systems that support the creation of loosely coupled applications in heterogeneous and distributed environment. Service computing or engineering covers the entire lifecycle of services that include: modeling, creation, realization, deployment, publication, discovery, composition, delivery, collaboration, monitoring, adaptation, optimization, and management. In this paper we propose an architecture for dynamic composition of web services that is guided by live testing technique. The main focus is on the framework and composition requirements.

Keywords: Web services · Dynamic composition · Live testing · Runtime monitoring · Runtime trace analysis · Architecture framework

1 Introduction

Service-Oriented Architecture (SOA) consists of a wide range of standards, specifications and tools. Standards from W3C and OASIS have been established to manage web service lifecycles and service-client communication. These standardized protocols and proposed languages played key roles in the development of web services. The challenges in service composition are mainly related to live real-time dynamic service composition: (1) efficient web service discovery; (2) efficient selection of basic services at run time; (3) on the fly verification and validation of composite web service; (4) user preferences driven discovery of basic web services; (5) context aware composition of services; (6) end to end quality proprieties of composed services: determination and guaranty; (7) dynamic assessment of quality of service; (8) verification of web service selection for transactional composition; (9) billing and pricing management; (10) service

F. Belqasmi et al. (Eds.): AFRICATEK 2017, LNICST 206, pp. 129–139, 2018.
https://doi.org/10.1007/978-3-319-67837-5_13

management, monitoring and adaptation; (11) intellectual and ownership rights. In this paper we only address a subset of the listed challenges.

The paper has the following sections: background information, related work, requirements for dynamic composition of web services, architecture framework for dynamic composite of web services, and then the conclusion.

2 Background Information

Service composition is used for application integration. A service is a means to transfer information from one component to the other. It can transfer information such as a message. A service is operable from any type of platform or operating system. It is in itself flexible. It is well known that a service has the following set of principles to be followed: (1) standardized contract; (2) loose coupling; (3) abstraction; (4) service reusability; (5) service autonomy; (6) service statelessness; (7) service discoverability; (8) composability. It is also known that Service Oriented Architecture (SOA) has 6 core values: business, strategic goals, intrinsic inter-operability, shared services, flexibility, and evolutionary refinement. The remaining parts of this section present the life cycle of web service composition, and related of service composition, dynamic composition, live testing and testability, runtime monitoring, runtime verification and runtime trace analysis.

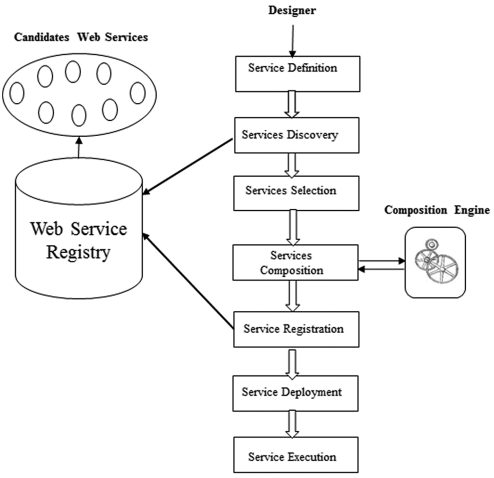

Fig. 1. The life cycle of services composition [30].

To better understand what is involved in composition of services, we introduce its lifecycle. The life cycle of web services composition is now well described in the literature [30]. It is composed of several phases (Fig. 1 shows the life cycle) that starts with a user's request for an added value service that doesn't exist as a standalone service. The request is forwarded to a service designer that will define the new service as a composition of existing services. Composition templates or design pattern can help.

This will trigger several activities that compose this life cycle such as services discovery, services selection, and composition of a service. An obtained composite service will be registered, and it will be deployed and executed as per the user's request.

To our knowledge there is no work addressing dynamic composition of services that is guided by live testing. In the following, we will address related work of all activities that are required by runtime composition of web services that is guided by live testing.

2.1 Dynamic Composition of Services

There are two types of service composition: (1) static that is done at design time [3]; and (2) dynamic that adapt to dynamic environment changes [1, 3, 42]. The authors of [4] offer a survey reviewing web composition. Paper [25] offers a framework for dynamic web service composition to different components. They use a graph-based approach to the web service composition. They take into consideration both functional and non-functional properties. The authors in [5, 6] identify a lifecycle for a web service composition. In each, a business aspect to identify the need for the composite web service is identified. In [2], there is a proposal for an approach that will render the composition of web services by rendering them self-aware of the context in which they are being composed. They intend to augment the semantics defining a web service in order to allow it to grow and adapt their behavior to a changing context. The authors have generated a UML activity diagram portraying a scenario to which an OWL-Ctx ontology representing their context, hence supporting their approach. In [17], context awareness is also in study for parallel multi-core components. This entails the possibility of executing multiple tasks in parallel which improves speed.

Nonfunctional requirements are important to service users. In [8], requirements are accessibility, availability, reliability, and performance. Additional requirements in [8] are mentioned such as execution price, execution duration and reputation. The requirements are non-functional characteristics that aim to ensure that both the client and service provider's experience with web services is up to scale. The authors in [7] have adapted a model they named *publish-find-bind* by integrating certification and verification transitions in relation to the quality of service. In [9] a concern related to security is raised in terms of authentication, authorization, confidentiality, cryptography, accountability, security administration, and non-repudiation.

Dynamic composition of web services palliates a known issue with static composition of web services: the inability to adapt to a changing context. This type of composition opens the path for live-testing as it pushes research towards self-awareness and context-awareness [10, 11, 27, 31]. Self-awareness of the web service would facilitate and optimize the dynamic composition due to the ability to satisfy fluctuating business requirements, in being aware of the new context, and integrating it. The challenge is the performance and the means for data storage. The amount of data needed for a service can be voluminous, and its performance could be affected. Making a context self-aware has been discussed and analyzed in many articles such as in [2, 10, 11, 17]. In [12], the authors discuss the design of an architecture to support and maintain self-awareness and enable live-testing through dynamic composition of web services.

2.2 Live Testing and Testability Related Work

Testing is the most used validation technique. It is expensive and technically challenging. A lot has been done on software testing. In this paper, the focus is on testing web services for their composition [43]. Live testing/runtime testing is an emerging research topic that is necessary to update at runtime composition of services with high availability requirements. Runtime testing is performed during the normal operation of services within its execution environment [38]. Most of the runtime testing approaches are platform dependent. For general work on runtime testing, concepts can be found in Brenner et al. publication [35]. All the proposed runtime testing techniques assume a predefined set of test cases. The research question is how to select a subset of test cases to be applied in relation with an observed event, a property violation, or an identified context. Runtime testing is applied in areas of embedded systems as a built-in-test [37], finance and e-commerce, as well as telecommunication systems. Gonzalez et al. [36] defined prerequisites to runtime testing, such as test sensitivity and test isolation notions, which are also important for runtime composition. A most recent work on safe and efficient runtime testing of distributed systems has been proposed by Lahami et al. [38] where the distribution of monitors and tester is the main focus. This work uses a set of predefined test cases. This limits their ability to generate additional test cases to adapt to unforeseen bugs and to diagnose their causes. The originality of our proposed framework is the architecture that allows the generation of test cases on the fly. The generation algorithm is guided by runtime trace analysis and objectives.

2.3 Runtime Monitoring, Verification and Trace Analysis

Early work on software monitoring focused on off-line monitoring, where data is collected at runtime and analyzed off-line. Runtime verification is an approach that allows insertion of services properties into the systems' code and verify them at execution time. The objective is to verify that services' properties are not violated, and the traces satisfy the composite service specification. The research challenges of runtime monitoring and runtime verification include constructing efficient monitors under the assumption of resources sharing with the observed service. The main concepts and foundations of runtime monitoring can be found in Viswanathan's PhD thesis and a survey [32, 33]. Survey and future challenges of distributed monitoring are addressed in [34]. Model based runtime verification/monitoring has been addressed in several papers listed in Zhao's early work [41].

Runtime trace analysis faces the same challenges as runtime monitoring and verification activities. They both require a valid specification of expected behavior, computation and memory resources. In addition, they should not interfere with the service and degrade its performance. Trace analysis is more specific to service behavior with the intention to detect nonconformance to the specification. Very often, they differ in the level of granularity in data collection, a trace analyzer is high level granularity in comparison to a monitor. To the best of our knowledge, a runtime composition of distributed services supported by guided live testing has not been studied yet.

3 An Architecture Framework for Dynamic Composite Web Services

We propose an architecture framework for live testing guided dynamic composition based on a broker (see Fig. 2). These types of architecture were used for QoS based selection of services for composition [37, 39]. We propose that the composition broker not only guides the client in offering web services composition, but also handles the guided dynamic composition itself and the verification aspect of the composition. In this architecture, modules for dynamic web service composition and live testing are located on the same system for efficiency and performance reasons. Depending on a type of service and the resources available at service provider, as well as the flexibility in instrumenting the service, the monitor can be either an insertion of monitoring code or a mobile agent attached to the monitored service. If the provided platform does not allow such insertion, then observation, monitoring and data collection are performed at the broker side. Runtime monitoring also helps in acquiring self-awareness during dynamic composition which allow adaptation of services.

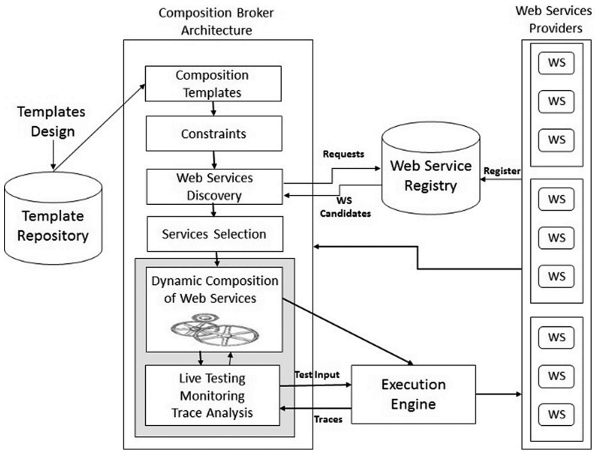

Fig. 2. Broker based architecture for live test guided dynamic composition of web services.

In this architecture we assume that composition templates and constraints exist or can be created by the service developer. The issue that we would like to address is to be able to abort or rollback the dynamic composition if needed after a step in testing. For this purpose, the dynamic composition needs to be guided (validated) by live testing.

The dynamic composition of services guided by live testing algorithm is depicted below (see Fig. 3). We use a composition algorithm that takes a pair of web services that have been selected. Then, monitor the states of services, collect information for possible rollback if needed. Afterwards, test web services invocations in sequence with a set of available inputs. In doing this, we use testing strategy, collect the outputs, and return a verdict by comparing the observed output with the expected one. If the comparison holds, the live testing module will generate the next invocation (composition).

If the observed output is different from the expected one, then the composition can simply abort if there is no damage or rollback, and move to the next pair of services to be composed. Monitoring and testing are under timing constraints to allow real-time dynamic composition.

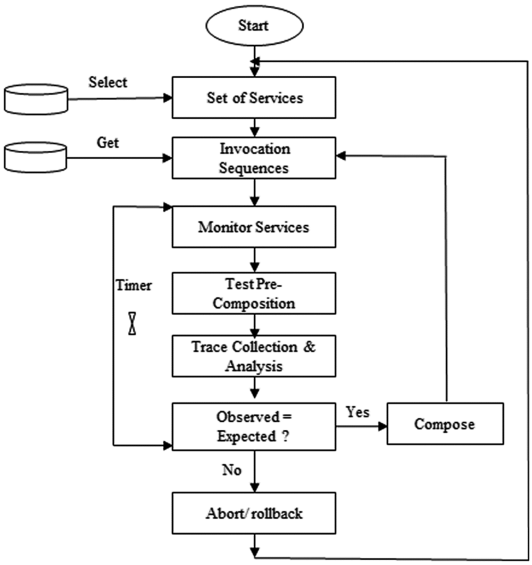

Fig. 3. General dynamic composition guided by live testing algorithm.

In the following we define the components of the architecture and address some challenges of live testing guided dynamic composition of web services.

3.1 Challenges of Live Testing Guided Dynamic Web Services Composition

The open problems are in user driven creation and composition of services in the context of dynamic environments. Usability and real-time discovery, selection, verification and composition of services are important issues. Scalability is becoming an important issue when dealing with large number of services. Proprieties of services are hidden problems such as accuracy, reliability, security and QoS that need to be reviewed and dealt with in the context of dynamicity and real-time composition of services. The following will describe the set of activities that are needed and is specific to live test guided dynamic composition of services.

3.2 Live Composition of Services

The aim is to develop techniques that compose services without stopping their operation. This type of composition faces several challenges, such as their readiness to be composed, how to capture information about services that are in operation and need to

be composed, how to capture and isolate components that need to be composed, how to analyze the impact of the composition on services' behavior, properties, and performance, and most importantly, how to foresee it before each step of the composition, how to verify services proprieties in run time, and how to rollback a composition that might impact a services' reliability, security and correctness. Live composition needs a pre-composition phase that is dedicated for instrumentation during the operation of the services to be composed. This step is to be done at runtime. Instrumentation is necessary for information acquisition, run time monitoring or verification, run time testing, and test result analysis for live guided testing.

Live composition is based on specification requires modeling at different levels of abstraction. It requires investigating both cases: (1) shared computation infrastructure between composition and services infrastructures; (2) collaborative computation infrastructure for composition, verification, testing and trace analysis to achieve the required performance and efficiency for runtime activities. The composition of services is a recursive activity that is performed by selecting, monitoring, isolating, pre-testing, integrating, verifying and post-testing all the services and their integration. To perform all the mentioned activities, scheduling of activities plays an important role. Contributions will be in regards to the development and construction of a composition engine of distributed services that is supported by a guided runtime testing, runtime trace analysis and diagnostics.

3.3 Live Testing of Services Composition

The objective of live testing is to be able to test services while they are in operation. This technique is complementary to traditional active and passive testing. Live testing requires instrumentation, monitoring and test results analysis. It can be used in conjunction with runtime verification of proprieties. The published research considers an existing set of test cases and a selection taking place during the execution of the service. This is in order to avoid interference between testing activity and the normal operations of the service and to minimize the impact of the performance. In this case, test cases are generated offline following testing method and test generation algorithm. The challenge in this type of testing is the selection of a minimal subset of suitable test cases that will reveal composition related bugs. This technique requires isolation of the services that are to be tested. A sensitivity analysis is carried out and test architecture is known or deduced. Guided Live Testing generates useful test cases on the fly and applies them in anticipation manner on selected and isolated services. Guided means that the generation of next inputs are based on previous inputs and the analysis of outputs (traces).

The generation algorithms have important performance requirements. Contrary to offline test generation algorithms, the guided live test generation algorithms should have strategies on what to test and why. How to perform distributed testing at runtime while avoiding interferences with the services under test. In addition, the challenge in performance and efficiency will be linked to what to drop and what to test. Guided live test algorithm uses objectives to generate suitable test cases that will detect interaction, and violations of specification and properties. Guided live testing is useful in the case of runtime composition of services, it helps cope with unexpected events and conditions

during the composition, and it helps locate the bug by generating additional test cases and performing runtime trace analysis. It can also help rollback the composition if needed.

3.4 Runtime Monitoring

Runtime monitors construction is still a research challenge. It is related to the instrumentation and trace collection at certain locations of the services. The monitors share resources with the monitored services, and have some impact on resource utilization in terms of computation and memory consumption; they may degrade the services performance. The additional challenge is the selection of the location of monitors and their choreography, coordination and possible migration. Monitors can be seen as mobile agents that can migrate according to the tasks and the locality of observation to be gathered. Monitors may be constructed with sort of contract to be preserved. We are going to build on our strengths in verification of mobile agents and their commitments with dynamic setting. Runtime verification of additional properties is known as lightweight verification technique that verifies whether a service execution satisfies or violates a set of given properties. It is partial to avoid state space explosion problem. It also has the capacity to scale with the number of proprieties to verify. Normally, runtime verification is implemented by constructing specific monitors.

3.5 Run-Time Test Results Analysis

Also known as oracle is an analyzer of test results with the intention to check whether the service under test has behaved correctly on a particular test case execution. It is a comparison between the observed behavior and a specified behavior that is assumed to be correct. Runtime test result analyzer is more complex due to time consumption and interferences with services operations and performance. For that particular reason runtime analyzers are very often partial. The challenges are many: how to filter only useful and relevant traces to analyze, existence of a specification, test architecture, computation resources and memory, execution efficiency, and time budgeting strategies. In this proposed research we will address the specific case of live guided test case generation, where a test input is generated based on the analysis of previous traces (outputs) and a model specification. This implies that trace analysis has to be partial and performed at runtime. In addition, not all traces are important to analyze, a selection of what to analyze and it dependency and impact on next operation of the service is important and challenging.

4 Conclusion

The objective of this is the development of broker-based architecture that allow QoS/guided live test dynamic composition of web services. The goal of the broker is to support live composition of web services with QoS. In addition, we addressed the problem of web services testing. The requirements for the implementation of the

proposed architecture are the development of (1) efficient web services discovery methods; (2) dynamic web service selection with QoS constraints and user preferences such as price; (3) dynamic composition and on the fly verification; (4) reliable execution of composite services; (5) approaches and tools for dynamic composition of mobile services; (6) develop a framework that will allow users to create mobile services and compose value added services on the fly; (7) develop a framework for user driven creation and composition of mobile services that include cloud computing (for the back end).

References

1. Ordonez, A., Alcázar, V., Corrales, J., Falcarin, P.: Automated context aware composition of Advanced Telecom Services for environmental early warnings. Expert Syst. Appl. **41**, 5907–5916 (2014)
2. Furno, A., Zimeo, E.: Context-aware Composition of Semantic Web Services. Springer Science + Business Media, New York (2014)
3. Khadka, R., Sapkota, B.: An evaluation of dynamic web service composition approaches, pp. 67–79 (2010)
4. Sheng, Q.Z., Qiao, X., Vasilakos, A.V., Szabo, C., Bourne, S., Xu, X.: Web services composition: a decade's overview. Inf. Sci. **280**, 218–238 (2014)
5. Aslam, M.A., Shen, J., Auer, S., Herrmann, M.: An integration life cycle for semantic web services composition. In: 11th International Conference on Computer Supported Cooperative Work in Design. IEEE, pp. 490–495 (2007)
6. Moghaddam, M., Davis, J.G.: Service selection in web service composition: a comparative review of existing approaches. In: Web Services Foundations. Springer New York, pp. 321–346 (2014)
7. Ran, S.: A model for web services discovery with QoS. ACM Sigecom Exchanges **4**, 1–10 (2003)
8. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q. Z.: Quality driven web services composition. In: Proceedings of the 12th International Conference on World Wide Web, pp. 411–421. ACM (2003)
9. Kuyoro Shade, O., Frank, I., Awodele, O., Okolie Samuel, O.: Quality of service (Qos) issues in web services. IJCSNS Int. J. Comput. Sci. Netw. Secur. **12**(1), 94–97 (2012)
10. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: International Symposium on Handheld and Ubiquitous Computing, pp. 304–307. Springer, Heidelberg (1999)
11. Hafiddi, H., Baidouri, H., Nassar, M., Kriouile, A.: An aspect based pattern for context-awareness of services. Int. J. Comput. Sci. Netw. Secur. **12**(1), 71–78 (2012)
12. Mustafa, F., McCluskey, T.L.: Dynamic web service composition. In: 2009 IEEE International Conference on Computer Engineering and Technology, ICCET 2009, vol. 2, pp. 463–467. IEEE (2009)
13. Qiao, M., Khendek, F., Serhani, A., Dssouli, R., Glitho, R.: Automatic QoS adaptation for composite web services. In: IEEE International Conference on IIT, pp. 180–184 (2008)
14. Valipour, M.H., AmirZafari, B., Maleki, K.N., Daneshpour, N.: A brief survey of software architecture concepts and service oriented architecture. In: IEEE International Conference on ICCSIT, pp. 34–38 (2009)
15. Sun, H., Wang, X., Zhou, B., Zou, P.: Research and implementation of dynamic web services composition. In: Advanced Parallel Processing Technologies, pp. 457–466 (2003)

16. Benatallah, B., Sheng, Q., Dumas, M.: The self-serv environment for web services composition. IEEE Internet Comput. **7**, 40–48 (2003)
17. Kessler, C., Löwe, W.: Optimized composition of performance-aware parallel components. Concurrency Comput. Practice Exper. **24**, 481–498 (2011)
18. Weigand, H., van den Heuvel, W.J., Hiel, M.: Rule-based service composition and service-oriented business rule management. In: Vanthienen, J., Hoppenbrouwers, S. (eds.) Proceedings of the International Workshop on Regulations Modeling and Deployment (ReMoD 2008), pp. 1–12. ACM (2008)
19. Char, A., Mezini, M.: Hybrid web service composition: business processes meet business rules. In: Proceedings of the 2nd International Conference on Service Oriented Computing, pp. 30–38. ACM (2004)
20. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.: eFlow: a platform for developing and managing composite e-services. Technical Report HPL-36, HPL (2000)
21. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.: Adaptive and dynamic service composition in eFlow. In: Advanced Information Systems Engineering, pp. 13–31. Springer, Heidelberg (2000)
22. Aggarwal, R., Verma, K., Miller, J., Milnor, J.: Dynamic Web Service Composition in METEOR-S. Technical report, LSDIS Lab, Univeristy of Georgia, Athens (2004)
23. Pires, P., Benevides, M., Mattoso, M.: Building reliable web services compositions. Web, Web-Services, and Database Systems, pp. 59–72 (2002)
24. Pires, P.F.: WEBTRANSACT: A Framework for Specifying and coordinating reliable web services compositions. Technical Report ES-578/02, Federal University of Rio De Janerio (2002)
25. Lécué, F., Silva, E., Ferreira Pires, L.: A framework for dynamic web services composition. In: Emerging Web Services Technology II, pp. 59–75 (2007)
26. Silva, E., Ferreira Pires, L., van Sinderen, M.J.: Supporting dynamic service composition at runtime based on end-user requirements. In: Workshop at the International Conference on Service Oriented Computing (ICSOC) 2009, Stockhome, Sweden, pp. 22–27 (2009)
27. Ordonez, A., Alcázar, V., Corrales, J., Falcarin, P.: Automated context aware composition of advanced telecom services for environmental early warnings. Expert Syst. Appl. **41**, 5907–5916 (2014)
28. Tiwana, A., Ramesh, B.: E-services: problems, opportunities, and digital platforms. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (2001)
29. Casati, F., Ilnicki, S., Jin, L.J., Krishnamoorthy, V., Shan, M.C.: An open, flexible, and congurable system for e-service composition. Technical Report HPL-2000-41, HPL (2000)
30. Sheng, Q.Z., Qiao, X., Vasilakos, A.V., Szabo, C., Bourne, S., Xu, X.: Web services composition: a decade's overview. Inf. Sci. **280**, 218–238 (2014)
31. Han, S.N., Lee, M.G., Crespi, N.: Context-aware service composition framework in web-enabled building automation system. In: International Conference on Intelligent in Next Generation Networks, pp. 128–133 (2012)
32. Viswanathan, M.: Foundations for the run-time analysis of software systems. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA (2000)
33. Leucker, M., Schallhart, C.: J. Logic Algebraic Program. A brief account of runtime verification **78**, 293–303 (2009). Elsevier
34. Goodloe, A., Pike, L.: Monitoring Distributed Real-Time Systems: A Survey and Future Directions. NASA/CR, Virginia 23681-2199 (2010)
35. Brenner, D., Atkinson, C., Hummel, O., Stoll, D.: Strategies for the run-time testing of third party web services. In: IEEE International Conference on Service-Oriented and Applications (SOCA 2007), pp. 114–121 (2007)

36. González, A., Piel, E., Grob, H.G.: Architecture support for runtime integration and verification of component-based Systems of Systems. In: ASE Workshops, pp. 41–48 (2008)
37. Suliman, D., Paech, B., Borner, L., Atkinson, C., Brenner, D., Merdes, M., Malaka, R.: The MORABIT approach to runtime component testing. In: 30th COMPSAC, pp. 171–176 (2006)
38. Lahami, M., Krichen, M., Jmaeil, M.: Safe and efficient runtime testing framework applied in dynamic and distributed systems. Sci. Comput. Program. **122**, 1–28 (2016). Elsevier
39. Serhani, M.A., Dssouli, R., Hafid, A., Sahraoui, H.: A QoS broker based architecture for efficient web services selection. In: IEEE International Conference on Web Services (ICWS 2005), pp. 113–120 (2005)
40. Oh, SC., Lee, D., Kumara, SRT.: Effective web service composition in diverse and large-scale service networks. IEEE Trans. Serv. Comput. **1** (2008)
41. Zhao, Y., Oberthür, S., Kardos, M., Rammig, F.J.: Model-based runtime verification framework for self-optimizing systems. Electr. Notes Theor. Comput. Sci. **144**, 125–145 (2006)
42. Lemos, A.L., Daniel, F., Benatallah, B.: Web service composition: a survey of techniques and tools. ACM Comput. Surv. (CSUR), vol. 48 (2016)
43. Bertolino, A.: Software testing research: achievements, challenges, dreams future of software engineering. In: FOSE 2007. IEEE (2007)