Towards Runtime V&V for Service Oriented Architectures

Cristiana Areias[1, 2], Nuno Antunes[2], João Cunha[1], Marco Vieira[2]

[1]CISUC, Polytechnic Institute of Coimbra
Coimbra – Portugal
cris@isec.pt, jcunha@isec.pt

[2]CISUC, Department of Informatics Engineering
University of Coimbra, Coimbra – Portugal
nmsa@dei.uc.pt, mvieira@dei.uc.pt

***Abstract*—The widespread use of SOAs and their specific characteristics raise new challenges for V&V practices. This paper presents some of these challenges and introduces Runtime V&V as a possible future solution.**

Keywords-services; SOA; validation; verification;

I. INTRODUCTION

Service Oriented Architectures (SOAs) are nowadays used in a wide range of organizations and scenarios, including in business-critical systems. These architectures consist of several interacting software resources (services) that are designed to support the information infrastructure of the organization [1]. These architectures present particular characteristics as complexity, extreme dynamicity, and a very large scale of composable components/elements and services. The forthcoming evolution is expected to exacerbate this trend even more, together with other evident facets, such as the needs for high mobility, high scalability, and high flexibility.

Complying with nowadays organizations' requirements demands for deployment and maintenance of trustworthy dynamic service-based software systems, which naturally results in the superposition of the design and runtime phases, thus imposing the need for a **Verification and Validation** (V&V) paradigm shift. V&V is the process of checking if the system meets the specifications and fulfills the intended purpose [2]. Verification checks the conformance to the specification, while Validation is a quality assurance process used to get the evidences needed to assure that the system fulfills the intended requirements, including non-functional features such as security and dependability. Rigorous V&V forms the fundaments of critical applications and has been applied throughout the years in several domains such as the railways, automotive, or space.

Unfortunately this detailed checking of a system prior to its deployment does not fit a service oriented context where a multitude of services is being deployed, interconnected and updated, following software development approaches which favor rapid deployment and frequent updates of services. Some challenges of testing SOAs are presented and discussed in [3]. In [4] the authors present a tool for testing SOAs supported by a discovery algorithm that is able to trace the SOA evolution by automatically discovering the services that compose the architecture and the connections between them. The approach is then used in the context of a testing service for SOA validation [5] that is basically a composite service able to monitor SOA evolution and test the various services according to specific testing policies. However, there is no complete solution to address the problem of V&V in this environment.

The traditional lifecycle in V&V assumes a structured and highly documented software or system development process that allows gathering the required quality evidences, and presumes that the system does not evolve after deployment (i.e., the structure is stable over time). This represents a serious problem, as there are no V&V methods, tools and processes that can cope with the dynamic nature of service oriented architectures, as well as with many other prominent features of these systems.

To overcome this problem new V&V approaches are necessary during runtime, assuring the required quality of the dynamic and evolving service oriented architectures. **Runtime V&V** takes advantage of monitoring services and infrastructures that will support the runtime assessment of the system through the collection of measurements for quantitative analysis of security and trustworthiness.

II. CHALLENGES IN V&V FOR SOAS

The following paragraphs summarize the main characteristics of complex, high-demand critical services, which open the new challenges for V&V and represent key issues that must be solved in order to assure trustworthy dynamic service based software systems.

Incremental software release development style: the development of open, large-scale, dynamic service oriented systems, mostly based on a large number of successive software releases, creates new problems for V&V approaches, as **repeating the entire V&V process** for each release is not cost nor time effective, and regression forms of V&V are simply not known yet.

Predominance **of agility in the software development methodologies:** large-scale service oriented systems are designed more and more using agile software development methodologies (as opposed to well structured software development processes used in systems where V&V is traditionally applied), which are characterized by evolving requirements, incremental software releases, less formalization and less detailed documentation. This makes traditional V&V useless, as known V&V processes rely on well defined, well structured, and well documented requirements and software specifications, demanding the development of new **V&V methods that cope with the features of agile development processes and allow traceability to evolving requirements.**

Dynamic runtime composition of services: the highly dynamic nature of complex services that require constant adaptation to changes in the environment and demand online reconfiguration through runtime deployment and composition of services makes traditional V&V ineffective, as actually the system structure is changing all the time and has no fixed boundaries. This is a tremendous research challenge that demands **new concepts of V&V for dynamic and**

evolving systems, requiring an infrastructure for service and/or system monitoring and measurement, and allowing the collection of the information that is required for the continuous verification and validation (without impacting the normal operation of the system).

III. RUNTIME V&V FOR SOAS

Our goal is to develop a **Runtime V&V** approach that will continuously monitor and assess the services of the system. This way it is possible to evaluate if the behavior deviates from its specification and, in the case this happens if it represents a threat to the system. Applying V&V techniques in a service-oriented environment can be a hard and critical task because services are running and any changes can cause a general failure, not only in our own services but also in third-party services which we do not have control. Fig. 1 portrays our approach, interacting with a SOA. This approach consists of the following modules:

Specification: services are typically described by specifications that contain functional and non-functional aspects. The **Input** specifications are the starting point of the **Architecture**, which will be updated by the **Monitoring** module. Also, in agile software development methodologies, where specifications can evolve, additional Input may be provided. Whenever it is possible, past V&V information must be added to the architecture in order to be reused in future V&V plans, improving the efficiency of the process.

Monitoring: the **Gatherer Agent** monitors the target system getting information (represented by ⋘) not only from individual services but also from the execution environment Meanwhile, if the **Updates Checker** detects changes in the system, the V&V module is notified in order to perform Runtime V&V. Other usages for the information gathered are out of scope of this work.

V&V: this module starts by evaluating which techniques to apply in the changed components through a set of metrics producing a **Plan**. Several factors must be considered: 1) cost involved; 2) complexity; 3) types of access to the service: under control, partially under control or within-reach; 4) the type of features, functional or not, etc. Previous information is necessary to improve the efficiency of the process by reusing past V&V activities and to apply regression testing. Many times, some of the traditional V&V techniques as code inspection, formal methods and testing cannot be applied. Finally, the **Executor** performs the planned activities and, depending on the dynamicity of the environment, decides if the plan must be done with or without user intervention. Techniques as virtualization or stubs are available to avoid system damage during this phase. V&V activities executed must be added in the architecture for future use.

Results: the execution produces a set of results that will be stored and analyzed. If a faulty component is detected, an intervention to the system must be executed. A component may also be considered suspect (although not faulty) and in this case, alerts are raised. This may require manual intervention or can be done automatically, if the SOA is sufficiently well prepared for self-reconfiguration of the systems. These results allow the stakeholders to determine the level of confidence in the system over the time.

IV. CONCLUSION

Runtime V&V for SOAs should follow new paradigms to cope with the challenges created by SOAs. Instead of following the traditional life cycle in V&V, assuming a structured and highly documented development process, it applies V&V in a dynamic fashion, taking advantage of iterative monitoring services and infrastructures that will support the runtime assessment.

ACKNOWLEDGMENT

This work has been partially supported by the project "CErtification of CRItical Systems" (http://www.cecris-project.eu/, CECRIS), Marie Curie Industry-Academia Partnerships and Pathways (IAPP) number 324334, within the context of the EU Seventh Framework Programme (FP7).

REFERENCES

[1] A. Singhal, T. Winograd, and K. Scarfone, "Guide to Secure Web Services: Recommendations of the NIST," *Report, NIST, US Department of Commerce*, pp. 800–95, 2007.

[2] A. Abran, P. Bourque, R. Dupuis, J. Moore, and L. Tripp, *Guide to the Software Engineering Body of Knowledge - SWEBOK*. IEEE Press, 2004.

[3] G. Canfora and M. Di Penta, "Service-oriented architectures testing: A survey," *Software Engineering*, pp. 78–105, 2009.

[4] A. Ceccarelli, M. Vieira, and A. Bondavalli, "A Service Discovery Approach for Testing Dynamic SOAs," in *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW)*, 2011, pp. 133 –142.

[5] A. Ceccarelli, M. Vieira, and A. Bondavalli, "A Testing Service for Lifelong Validation of Dynamic SOA," in *2011 IEEE 13th International Symposium on High-Assurance Systems Engineering (HASE)*, 2011, pp. 1 –8.

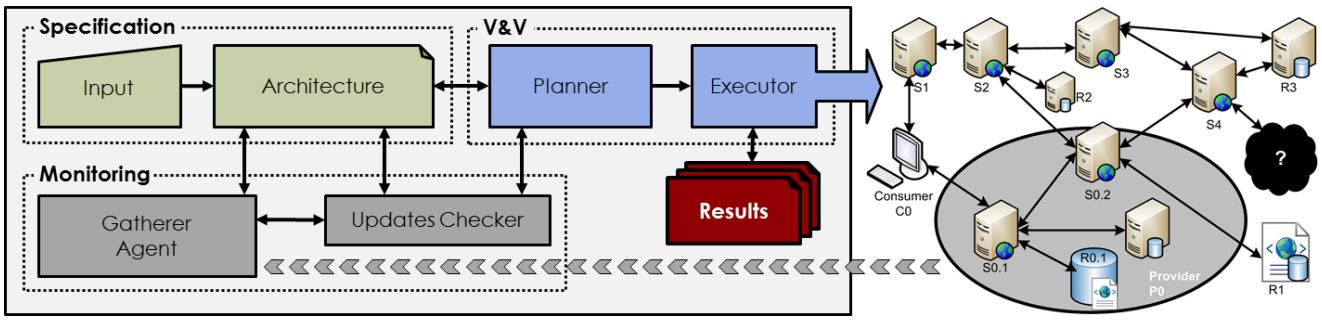

Figure 1. Runtime V&V interacting with a simple Service Oriented Architecture.