Do we need new strategies for testing Systems-of-Systems?

Vânia de Oliveira Neves
Universidade Federal de Juiz de Fora
Juiz de Fora, Brazil
vania@ice.ufjf.br

Antonia Bertolino
Consiglio Nazionale delle Ricerche
Pisa, Italy
antonia.bertolino@isti.cnr.it

Gugliemo De Angelis
Consiglio Nazionale delle Ricerche
Rome, Italy
guglielmo.deangelis@iasi.cnr.it

Lina Garcés
Universidade de São Paulo
São Carlos/SP, Brazil
linamgr@icmc.usp.br

ABSTRACT

This paper overviews the main Systems-of-Systems (SoS) characteristics that can impact on their verification, validation and testing (VV&T). Furthermore, it addresses technical, conceptual, social and organizational challenges, discusses which existing approaches of VV&T can be used for SoS, and points out future research in the field.

KEYWORDS

Systems-of-Systems, testing, validation

ACM Reference Format:
Vânia de Oliveira Neves, Antonia Bertolino, Gugliemo De Angelis, and Lina Garcés. 2018. Do we need new strategies for testing Systems-of-Systems?. In *SESoS'18: SESoS'18:IEEE/ACM 6th International Workshop on Software Engineering for Systems-of-Systems (, May 29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3194754.3194758

1 INTRODUCTION

In recent times, software systems perspective has changed from monolithic, centrally managed, and with well-defined scope, to large scale, complex, distributed, and with dynamically defined scope. Moreover, new business demands require the integration of existing software systems to achieve missions that cannot be addressed by any of them in isolation. This context has brought out the concept of Systems-of-Systems (SoSs), in which a set of independent systems cooperate to achieve broader missions than those they individually perform. An SoS mission is an operational goal and defines the capabilities required from its constituent systems [26]. SoSs concept emerged in the military domain, however, they are now applied in other domains like intelligent transports, medical assistance, and emergency management [23, 24].

SESoS'18, May 29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5747-0/18/05…$15.00
https://doi.org/10.1145/3194754.3194758

Several definitions of SoSs have been proposed (see, e.g., [19]). Maier [23] defines an SoS as an assemblage of components (referred as the *constituent* systems) that possess **operational independence** and **managerial independence** from the SoS. On its side, the SoS exposes features like **evolutionary development**, **emergent behavior**, and **geographical extension** [23, 24]. The above mentioned characteristics of SoSs and their constituents impact the entire SoS development process, hindering verification, validation and testing (VV&T) activities [2, 10, 11, 16, 22]. However, we found that some of the challenges impacting on VV&T of SoSs have already been addressed in other types of software-intensive systems such as, component-based systems, service-oriented architecture, and self-adaptive systems.

Moreover, SoSs are classified into four types: directed, acknowledged, collaborative, and virtual [23, 24]. **Directed** SoSs have a central authority that establishes an unilateral communication with the constituents, and orchestrates them; in **acknowledged** SoSs, a central entity establishes a bidirectional communication path and the constituent systems – which retain independent control and objectives – consult this entity to discover other constituents and services provided; in **collaborative** SoSs, the constituent systems must collaborate voluntarily to fulfill the established central objectives, and the central managing entity does not have coercive power to execute the system; in **virtual** SoSs, there is no centralized management authority and the constituents do not necessarily know each other. The final result for the SoS turns out as the sum of the partial results, without a clear intent or explicit collaboration between them.

Research in SoSs testing has been led by Systems Engineering researchers from the military field who conducted case studies to address specific issues for this domain [2, 10, 11]. Luna and Lopes [22] propose a framework for IVVT&E (*Integration, Verification, Validation, Test* and *Evaluation*) that unites different methodologies such as DoDAF (*Department of Defense Architecture Framework*), graphs theory and executable models. The PATFrame framework [14, 17] aims to predict when a test system needs to be adapted using the information learned during the testing process.

Few are the approaches for SoSs VV&T outside the military domain [3, 21, 27, 28]. Zapata et al. [28] present an approach using the basic path test to generate test case sets for an SoS. Constituent systems are modeled as nodes of a control flow graph and each decision point is modeled as if it were a conditional expression in the graph. Similarly, path testing has already been used earlier,

e.g. in [18], to model modules of a component-based system. Hsu and Huang [21] present an approach to designing test cases for SoSs in a way that minimizes the number of possible test cases during integration of the constituents. This approach was adapted from an earlier work proposed by the same authors for SOA applications [20]. Liang and Rubin [27] propose a strategy to evaluate the effectiveness in detecting failures using mutation testing for SoS policies. This approach does not generate test cases, it only measures the effectiveness of existing test cases. Arnold et al [3] propose an approach to SoS verification using Statistical Model Checking (SMS). For this purpose, SoSs goals are expressed using formal methods, SoSs and constituents are modeled using existing tools and then these models are simulated to produce traces for SMC.

In this opinion paper we discuss whether new challenges for VV&T of SoSs really exist that require novel techniques and methods, or instead, we just need new ways of combining and using existing approaches. Therefore, this paper aims to summarize the main technical, conceptual, and social challenges for SoSs VV&T, and to identify existing techniques from the literature that could be adapted to the context of SoS to address such challenges.

This paper is organized as follow: Section 2 presents the technological, conceptual, social and organizational challenges; Section 3 discusses what testing approaches proposed in other domains can be used for the problems identified in previous section; and, a final discussion is presented in Section 4.

2 CHALLENGES AND ISSUES IN SOS VV&T

As mentioned before, an SoS brings several advantages but with them also consequences that can impact on the VV&T activities in technical, conceptual, social and organizational aspects.

Due to **operational independence**, a constituent system can operate independently and perform its own functions even if it is disassembled from the SoS. As a consequence, it can have several functions, of which only a few are used to fulfill the SoS mission. If the constituent system does not have enough quality this could lead the SoS to fail. How can we apply unit testing with good performance? The constituent system may be very large and the supplier can provide numerous test cases, which test cases should be selected and how to select them? What would be an appropriate adequacy criterion for a constituent system? The answer to such questions will depend on the SoS under consideration: a constituent system may have achieved a high structural coverage, for example, but still the testing did not cover the very functions used by the SoS. Operational independence also may cause integration and SoS issues since the constituent systems can enter or exit the SoS, meaning that the services offered may vary in terms of functions and lead to other subsystems having to be reconfigured. This **interdependence between constituent systems** is presented by Nielsen et al. [24] as one of the eight SoS dimensions.

Managerial independence refers to the ability of the constituent systems to be managed and developed independently even while being part of the SoS. Hence, it is difficult to synchronize the activities of their life cycles; this brings **interoperability issues** since the integration cannot be direct due to, among others, different protocols for transmission and request of data and services; there is no guarantee of their availability or that all constituent systems are ready to perform a given task at any given time [11, 23]. According to Gonzalez et al. [16], when a constituent system enters the SoS, its collaboration must be accepted by the other systems and, in addition, there may be restrictions that allow or prevent the use of information coming from other constituents or pass this information to other systems. This raises integration questions like: how to ensure the right information was transmitted to the right systems? How to **orchestrate the testing**? Also, how to perform integration testing considering these features?

Evolutionary development causes constant updates or evolution of SoS requirements and can lead to the lack of a permanent state in SoS [24]. Thus, we need to continuously evaluate if no regression problem has been brought by the evolved component [16, 22]. The problem is that if the number of interactions is significantly high, which is most often true in the SoS context, a very large number of test cases has to be generated involving a great effort and cost. So, the question here is **when to perform VV&T**? Carrying it out each time an SoS changes would be expensive but, on the other hand, changes can incur risks for different participants.

As mentioned earlier, SoS allows to achieve purposes that would not be feasible individually by the individual systems. This is what is called the **emergent behavior** that may bring consequences such as inappropriate or unexpected behavior that affects the testing. Challenges here include how to detect and destroy in a fast and agile way this inappropriate behaviour. Further, how to assure that the constituent systems behavior is aligned with the SoS mission? Nielsen at al. [24] also point **dynamic reconfiguration** as an SoS dimension since the SoS can change its composition depending on the conditions and on the context, typically at runtime and without planned intervention. This poses additional challenges like how to assure that this is the right configuration? Could this configuration result in an inappropriate emergent behavior?

In addition to the above technological challenges, there are also conceptual challenges that need to be addressed. In particular, we propose to revisit the next five fundamental concepts of SoSs testing:

(i) What would be a test environment? The test environment may involve other aspects transcending the VV&T activities [10] as people, infrastructure and legislation. For some SoSs it is difficult, or even impossible, to simulate the reality, thus we need to conduct testing in the field. How to do that? For instance, suppose a traffic control SoS in which the constituent systems are autonomous vehicles. How to ensure that a failed test does not cause any type of accident? Also, current laws do not allow a vehicle to travel without a driver on urban roads.

(ii) What should be tested? In many cases, an SoS has no requirements per se but high-level objectives that provide the basis for identifying constituent systems, developing the architecture, or recommending changes or additions to systems. Requirements are specified at the constituent systems level, not at SoS level [11]. This makes it difficult to define the scope of what should be tested.

(iii) What does "adequate testing" mean? Due to lack of defined requirements and the involvement of many stakeholders, each having their own expectations, it is not easy to identify test adequacy criteria for SoS. How can we decide that testing is enough? This question will need to be adapted to the stakeholder and to the context.

(iv) What is an SoS failure? An SoS consists of several stakeholders, for instance, SoS owner; SoS clients, if it provides new services, or constituent systems, otherwise; constituent systems stakeholders; and even other actors from the environment, such as a pedestrian hoping that the autonomous vehicle will stop when a semaphore is red. These stakeholders will have different interests and priorities, so one same execution can meet the expectations of a particular stakeholder but be insufficient or even prejudice another. Thus, what is a failed test? The oracle problem that is already tough in traditional systems becomes even more complex for SoSs.

(v) Who is the tester? SoSs are characterized by constituent systems belonging to different organizations. So, who is in charge for performing the test activities? Directed and acknowledged SoSs have a central management that could be the obvious responsible for these activities. But in collaborative SoSs the management is formed through a cooperation between the organizations of constituents systems; virtual SoSs have no form of management, which makes it impossible to determine a responsible. Moreover, if any defect is found, who will be the responsible for fixing it?

Finally, due to pervasiveness and context-awareness of SoSs, their quality or non-functional requirements become as important as, or even more important than, functional requirements. The design of an SoS taking into account quality requirements is actively studied at the CMU Software Engineering Institute (SEI) [15].

3 ADAPTING SOLUTIONS FROM OTHERS DOMAINS

Considering a systematic testing strategy, test activities are traditionally structured into three phases: unit, integration and system. Moving to SoSs, the unit testing phase can be intended as ensuring that the smallest SoS units have been sufficiently tested against their specifications. We consider here the constituent system as the smallest unit to be tested and we expect that its supplier performs all testing phases on it. After unit testing, the integration testing is performed, in order to investigate whether each constituent system cooperates adequately with its peers in order to guarantee the required (or emergent) properties. Finally, the SoS testing is performed on the assembly as a whole. This phase aims to verify that the SoS is actually fulfilling its mission.

Since the source code of the constituent system is not always provided by the supplier, the lack of such information makes it impossible to apply white-box test techniques. This problem is not new, but has been previously faced in testing of component-based systems, especially w.r.t. components developed by third parties and COTS, and in testing of service-oriented architectures. Solutions proposed to circumvent such limitations included the proposal to release along with a component its test metadata [13], the introduction of a certification third party [5], or even the establishment of a governance framework [9].

The problems brought by managerial independence feature are the same faced by service-oriented software systems. To deal with these problems, some of the authors have developed several approaches and tools such as [1, 4, 6, 8, 9]. For instance, the WS-TAXI tool [5] could be used/adapted to perform the service admission testing; the PUPPET tool [8], which creates mocks for external services, can be used/adapted to solve the problem of interdependence; the STG framework [9] could be used to ensure that data exchange between interfaces interacting with the system is properly performed. Besides these technologies, the Elastest platform [7] can be used for orchestrating the test cases. There are many other approaches proposed by other authors that we could have mentioned, we cited these just because we are of course more familiar with them.

Component-based systems face the problem of identifying proper adequacy criteria, and a conceptual framework has been proposed in [25]. In order to solve the operational independence issues, we may consider analogous criteria to those proposed for components. In an adapted version for SoS, we can consider *C-adequate-for-SoS* and *C-adequate-on-CS criteria*, where C refers to the test adequacy criterion; SoS to the system-of-system under test and CS the constituent system to be integrated.

Dynamic reconfiguration feature addresses similar challenges that were already faced by adaptive systems. To deal with this, methods and tools have been studied in this field. The state of the art and the main challenges can be found in [12].

4 DISCUSSION

This paper touched upon the main challenges faced when performing System-of-System testing and pinpointed some approaches in the literature that can be used or adapted to deal with them. Among the most promising strategies are those developed for testing of service-oriented software systems, component-based systems, and adaptive systems. Table 1 summarizes what we discuss in this position paper. *Characteristic* column lists the main characteristics of SoSs; *Consequences* column describes the key challenges related to the characteristic; *Testing issues* column, in turn, lists the major issues related to testing caused by these characteristics; *Phase* column points out which test phase these issues should be considered; and finally *Approaches from others domains* column indicates the existent solutions that could be considered/adapted to deal with these issues.

It should be noted that SoSs are highly complex systems, implying that experiments need to be conducted to validate the use of these approaches in this context. In addition, different types of SoSs should be considered in these experiments, since they can generate different challenges. Moreover, although the results related to adaptive systems testing can be used in SoSs, this field still presents many challenges to be addressed and, in this sense, issues related to emergent behavior need to be further investigated.

According to Table 1, there is no existing domain from which we could adapt approaches that could be used to deal with the issues raised by evolutionary development. Future research should consider the selection and prioritization of test cases each time SoSs evolve. Further, more research should be conducted in order to define when the validation and verification process should be performed. Our belief is that a strategy here will depend on how critical the SoS is.

Considering that missions may change at run-time and even cease to exist depending on the conditions and context they are in, strategies that validate and capture any unplanned behaviors of these missions are needed. One approach we are thus considering is to plan and guide testing for SoSs based on the missions to be accomplished: we call this *mission-driven testing* of SoS.

Table 1: Summary of SoSs features, issues related to testing and possible use of existing approaches

Characteristic	Consequences	Testing issues	Phase	Approaches from other domains
Operational independence	Interdependence between CS CS can have several functions, but few are used to fulfill the SoS mission	What should be tested? What would be an appropriate adequacy criterion for a CS? Which test cases provided by the supplier should be selected?	Unit	Component-based
Managerial independence	Life-cycle Interoperability Availability of a particular CS Reliability	Who is responsible for V&V? How to accept a constituent system? How to ensure the right information was transmitted to the right systems? How to orchestrate the testing?	Integration	SOA Component-based
Evolutionary development	Lack of a permanent state in SoSs	Number of test cases can be very large When to perform V&V?	Integration SoS	
Emergent behavior	Dynamic reconfiguration Inappropriate behavior Mission may change at runtime depending on conditions and context	How to detect and destroy in a fast and agile way? How to assure that CS behavior is aligned with the SoS mission? How to assure that they made the right change?	SoS	Adaptive-testing
Geographic distribution	Decentralized nature Concurrent	How should V&V be organized?	SoS	Concurrent testing

ACKNOWLEDGMENTS

This work has been partially supported by the GAUSS project (Italian MIUR, PRIN 2015, Contract 2015KWREMX), the H2020 European Project ElasTest (Grant N.731535), and the Brazilian Funding Agency FAPESP (Grant 2013/20317-9).

REFERENCES

[1] M. Ali, F. De Angelis, D. Fanì, A. Bertolino, G. De Angelis, and A. Polini. 2014. An Extensible Framework for Online Testing of Choreographed Services. *Computer* 47, 2 (Feb 2014), 23–29.

[2] N. B. Ali, K. Petersen, and M. V. Mantyla. 2012. Testing highly complex system of systems: An industrial case study. In *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. 211–220.

[3] A. Arnold, M. Baleani, A. Ferrari, M. Marazza, V. Senni, A. Legay, J. Quilbeuf, and C. Etzien. 2016. An Application of SMC to Continuous Validation of Heterogeneous Systems. In *Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques (SIMUTOOLS'16)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 76–85.

[4] C. Bartolini, A. Bertolino, S. Elbaum, and E. Marchetti. 2011. Bringing White-box Testing to Service Oriented Architectures Through a Service Oriented Approach. *J. Syst. Softw.* 84, 4 (April 2011), 655–668.

[5] C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini. 2009. WS-TAXI: A WSDL-based Testing Tool for Web Services. In *2009 International Conference on Software Testing Verification and Validation*. 326–335.

[6] A. Bertolino, G. De Angelis, S. Kellomaki, and A. Polini. 2012. Enhancing Service Federation Trustworthiness through Online Testing. *Computer* 45, 1 (Jan 2012), 66–72.

[7] A. Bertolino, A. Calabró, G. De Angelis, M. Gallego, B. García, and F. Gortázar. 2018. When the testing gets tough, the tough get ElasTest. In *Proceedings of the 40th International Conference on Software Engineering (ICSE '18)*. ACM, New York, NY, USA, 1–5.

[8] A. Bertolino, G. De Angelis, L. Frantzen, and A. Polini. 2008. Model-Based Generation of Testbeds for Web Services. In *Testing of Software and Communicating Systems*, Kenji Suzuki, Teruo Higashino, Andreas Ulrich, and Toru Hasegawa (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 266–282.

[9] A. Bertolino and A. Polini. 2009. SOA Test Governance: Enabling Service Integration Testing across Organization and Technology Borders. In *2009 International Conference on Software Testing, Verification, and Validation Workshops*. 277–286.

[10] J. Colombi, B. C. Cohee, and Chuck W. Turner. 2008. Interoperability test and evaluation: A systems of systems field study. *The Journal of Defense Software Engineering* (Nov 2008), 10–14.

[11] J. Dahmann, J. A. Lane, G. Rebovich, and R. Lowry. 2010. Systems of systems test and evaluation challenges. In *2010 5th International Conference on System of Systems Engineering*. 1–6.

[12] R. de Lemos, D. Garlan, C. Ghezzi, H. Giese, J. Andersson, M. Litoiu, B. Schmerl, D. Weyns, L. Baresi, N. Bencomo, Y. Brun, J. Camara, R. Calinescu, M. B. Cohen, A. Gorla, V. Grassi, L. Grunske, P. Inverardi, J. Jezequel, S. Malek, R. Mirandola, M. Mori, H. A. Müller, R. Rouvoy, C. M. F. Rubira, E. Rutten, M. Shaw, G. Tamburrelli, G. Tamura, N. M. Villegas, T. Vogel, and F. Zambonelli. 2018. Software Engineering for Self-Adaptive Systems: Research Challenges in the Provision of Assurances. In *Software Engineering for Self-Adaptive Systems III*, Rogério de Lemos, David Garlan, Carlo Ghezzi, and Holger Giese (Eds.). Vol. 9640. Springer, 3–30. https://doi.org/10.1007/978-3-319-74183-3_1

[13] M. M. Eler, A. Bertolino, and P. C. Masiero. 2013. Applying Structural Testing to Services Using Testing Interfaces and Metadata. *International Journal of Software and Informatics (IJSI)* 7, 2 (April 2013), 239–271.

[14] S. Ferreira, R. Valerdi, N. Medvidović, J. Hess, I. Deonandan, T. Mikaelian, and G. Shull. 2010. Unmanned and Autonomous Systems of Systems Test and Evaluation: Challenges and Opportunities. In *IEEE Systems Conference*. 15 pp.–.

[15] M. Gagliardi, B. Wood, and T. Morrow. 2013. *Introduction to the Mission Thread Workshop*. Technical Report. Software Engineering Institute, Pittsburgh, PA.

[16] A. Gonzalez, E. Piel, H. G. Gross, and M. Glandrup. 2008. Testing Challenges of Maritime Safety and Security Systems-of-Systems. In *Testing: Academic Industrial Conference - Practice and Research Techniques (taic part 2008)*. 35–39.

[17] J. T. Hess and R. Valerdi. 2010. Test and evaluation of a SoS using a prescriptive and adaptive testing framework. In *2010 5th International Conference on System of Systems Engineering*. 1–6.

[18] C. J. Hsu and C. Y. Huang. 2011. An Adaptive Reliability Analysis Using Path Testing for Complex Component-Based Software Systems. *IEEE Transactions on Reliability* 60, 1 (March 2011), 158–170.

[19] M. Jamshidi. 2009. *System of systems engineering: innovations for the 21st century*. John Wiley & Sons.

[20] Q. Liang and S. H. Rubin. 2007. Randomization in Searching for Composites of Software Components. In *2007 IEEE International Conference on Information Reuse and Integration*. 42–48.

[21] Q. Liang and S. H. Rubin. 2009. Randomization for testing systems of systems. In *2009 IEEE International Conference on Information Reuse Integration*. 110–114.

[22] S. Luna, A. J. Lopes, H. Y. S. Tao, F. Zapata, and R. Pineda. 2013. Integration, Verification, Validation, Test, and Evaluation (IVVT&E) Framework for System of Systems (SoS).. In *Complex Adaptive Systems (Procedia Computer Science)*, Cihan H. Dagli (Ed.), Vol. 20. Elsevier, 298–305.

[23] M. W. Maier. 1998. Architecting principles for systems-of-systems. *Systems Engineering* 1, 4 (1998), 267 – 284.

[24] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska. 2015. Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions. *ACM Comput. Surv.* 48, 2, Article 18 (Sept. 2015), 41 pages.

[25] D. S. Rosenblum. 1997. *Adequate Testing of Component-Based Software*. Technical Report. University of California, Irvine, CA.

[26] E. Silva, T. Batista, and F. Oquendo. 2015. A mission-oriented approach for designing system-of-systems. In *2015 10th System of Systems Engineering Conference (SoSE)*. 346–351. https://doi.org/10.1109/SYSOSE.2015.7151951

[27] W. Yun, D. Shin, and D. H. Bae. 2017. Mutation Analysis for System of Systems Policy Testing. In *2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS)*. 16–22.

[28] F. Zapata, A. Akundi, R. Pineda, and E. Smith. 2013. Basis Path Analysis for Testing Complex System of Systems. *Procedia Computer Science* 20 (2013), 256 – 261. Complex Adaptive Systems.