Simulation of Systems-of-Systems Dynamic Architectures

Wallace Manzano[1], Valdemar Vicente Graciano Neto[1,2], Elisa Yumi Nakagawa[1]

[1] Institute of Mathematics and Computer Science
University of São Paulo (USP) – São Carlos, Brazil

[2]Institute of Informatics
Federal University of Goiás (UFG) – Goiânia, Brazil

wallace.manzano@usp.br, valdemarneto@ufg.br, elisa@icmc.usp.br

Abstract. *Systems-of-Systems (SoS) combine heterogeneous, independent systems to offer complex functionalities for highly dynamic smart applications. Due to their critical nature, SoS should be reliable and work without interruption since a failure could cause serious losses. SoS architectural design can facilitate the prediction of the impact of failures due to SoS behavior. However, existing approaches do not support such evaluation. The main contribution of this paper is to present Dynamic-SoS, an approach to predict, at design time, the SoS architectural behavior at runtime to evaluate whether the SoS can sustain their operation. Results of our multiple case studies reveal Dynamic-SoS is a promising approach that could contribute to the quality of SoS by reliably enabling prior assessment of their dynamic architecture.*

1. Introduction and Background

Context. Software-intensive systems have been increasingly required to interoperate among themselves, communicating, exchanging, and using information exchanged[1]. Consequently, a distinct class of systems known as Systems-of-Systems (SoS)[2] has arisen. An SoS results from operationally and managerially independent software-intensive systems (called constituents) working together to fulfill complex missions (Maier 1998). SoS are often linked to critical domains, such as smart traffic control systems, crisis response management, and national defense systems.

Due to their inherent criticality, it is imperative that SoS are reliable, that is, their operation must be correct and happen according to what is expected, without causing harm or threatening people who use their services. To guarantee a reliable operation for SoS, it is important to establish strategies to maintain an SoS operation in progress, despite a high degree of dynamism, with new constituents joining and leaving the SoS at runtime (Heegaard and Schoitsch 2015). Besides, it is essential that SoS operation is not interrupted and that problems in one or more of their constituents are automatically solved by automatic and dynamic reconfiguration of the SoS architecture at runtime to substitute them and guarantee the feasibility of the SoS continuous operation.

Simulation techniques have been successfully used in software engineering (de França and Travassos 2016) to support the visualization of the systems' dynamic

[1] http://www.himss.org/library/interoperability-standards/what-is-interoperability

[2] For sake of simplicity, herein the acronym SoS will be used interchangeably to express both singular and plural.

behaviors. Simulations can anticipate, at design time, failures and behaviors that could potentially occur at runtime. Simulation of software architectures can be achieved by using a simulation formalism such as Dynamic Structure Discrete Event System Specification (DS-DEVS) to modeling behavior of systems by a table of inputs, outputs, and state transitions, as well as the coupling of these systems, as adopted by Bogado et al. (2017).

Problem. Despite the suitability of simulation formalism to predict behaviors with dynamic reconfiguration support, a complex task arises from ***guaranteeing precision in simulating SoS software architectures, i.e., simulating the behavior of multiple constituent systems and their relationship to each other, and to the environment at runtime while still considering the dynamic reconfiguration of the SoS constituent systems***. Existing simulation formalism lacks on precisely representing SoS software architectures, including the representation of different types of systems, environment representation, and dynamic reconfiguration. Despite the formalism supporting the dynamic architecture representation, the mechanism to implement it is arbitrarily chosen by the user. On the other hand, the pure adoption of architectural description languages (ADL) also suffers from limitations on the precise description of dynamism of an architecture composed by multiple systems and the impact of it on the provided behavior, as recognized by Guessi et al. (2015). The main research question addressed in this work is: *How to provide a means to assure that the SoS operation is trustworthy, i.e., will it be maintained on-the-fly, despite the inherent SoS dynamic architecture?*

Solution. To address this question, this paper presents Dynamic-SoS, an approach established by the undergraduate student for simulating SoS software architectures with dynamic reconfiguration support. Our approach uses a SosADL to DEVS model transformation (SosADL2DEVS) that automatically generates simulation models from SoS software architecture specifications. The main contributions of Dynamic-SoS were: (i) the proposition of a set of canonical dynamic reconfiguration operators for SoS software architectures; (ii) the extension of an existing model transformation by adding a dynamic reconfiguration controller (DRC) structure, which enables simulation models in DEVS to show and manage dynamic reconfigurations at runtime; (iii) the engineering of DRC itself, since such an idea could be used for other simulation formalism, as well; and (iv) the characterization of the process to assess SoS behaviors considering the associated dynamic architecture. We evaluated our approach through multiple case studies in three distinct domains: Flood Monitoring SoS, Smart Building SoS, and Space SoS. From a triangulation of the results obtained from these case studies, we concluded our approach successfully supports SoS simulation while accounting for dynamic architectures. As a result, our work may potentially be used to evaluate SoS dynamic architectures, enhancing SoS quality by enabling the visualization of possible problems that the SoS may exhibit besides predicting/anticipating SoS architectural behavior at runtime, thus evaluating whether the SoS can sustain their operation.

This paper is organized as follows: Section 2 presents Dynamic-SoS, Section 3 presents our evaluation and results, Section 4 summarizes the contributions of this work, and Section 5 contains final remarks and directions for future work.

2. Dynamic-SoS Approach

Dynamic-SoS is a model-based approach to automatically create SoS simulations with dynamic reconfiguration (Manzano et al. 2019). To support the dynamic reconfiguration, we established a set of dynamic reconfiguration operations based on the canonical set of changes initially proposed by Cavalcante et al. (2015). Cavalcante et al. observed single systems may have their architectures modified at runtime due to the creation of a new component, removal of a component, attachment of a component, and detachment of a component. For SoS software architectures, we reviewed such operators and defined four canonical operators to achieve dynamics in such architectures. In our approach, dynamic reconfiguration in an SoS software architecture is based on constituent addition, constituent removal, constituent replacement, and architecture reorganization. Our approach uses a SosADL to DEVS model transformation (SosADL2DEVS) that automatically generates simulation models from SoS software architecture specifications, based on an approach called ASAS (Approach to Support Simulation of SmArt Systems) (Graciano Neto et al. 2018a). However, ASAS does not support the simulation of dynamic architectures. Dynamic-SoS was established by adding the generation and monitoring of dynamic architectures for that SosADL2DEVS transformation produced by Graciano Neto et al. (2018a), as depicted in Figure 1. Moreover, we established a set of activities to systematize the use of Dynamic-SoS approach, as follows.

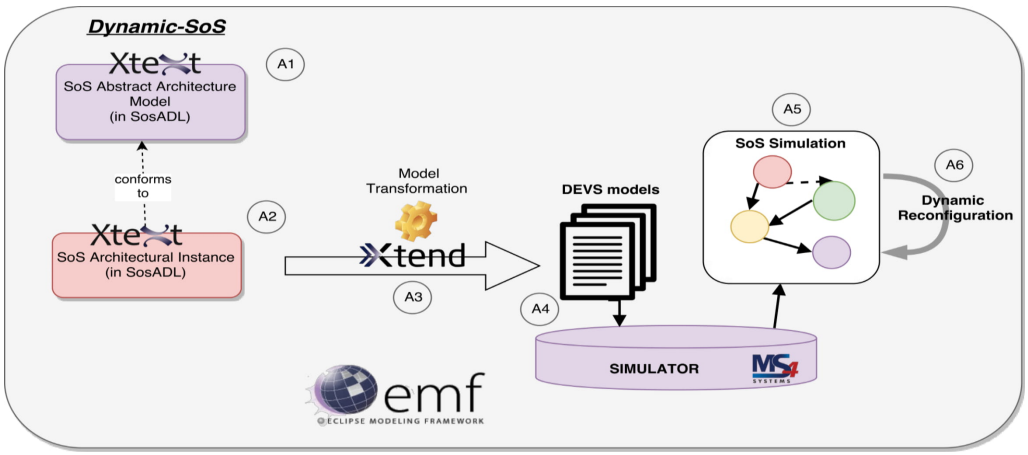

Fig. 1. Dynamic-SoS approach (Manzano et al. 2019; Graciano Neto et al. 2018a).

Activity 1 (A1). Definition of the SoS architecture by means of a coalition, expressing the policies and defining bindings among the constituents;
Activity 2 (A2). Generation of the initial architectural arrangement for the simulation, using a method defined by Guessi et al. (2016);
Activity 3 (A3). Execution of the model transformation on the concrete model (in SosADL) to automatically produce a DEVS simulation code with support for the dynamic reconfiguration of an SoS architecture;
Activity 4 (A4). Deployment, i.e., the process of managing files of the atomic and coupled models obtained as an outcome of the transformation, deploying them in the specific packages/directories of the project to be simulated in MS4ME[3] (the simulation environment);
Activity 5 (A5). Simulation execution that consists of launching the simulation; and

[3] http://www.ms4systems.com/pages/ms4me.php

Activity 6 (A6). Monitoring architectural reconfiguration using the dynamic reconfiguration operators, comprising the observation of the SoS dynamics at runtime, and registering the execution traces in log files for posterior analysis.

2.1. Dynamic reconfiguration controller

Dynamic-SoS is a programmed exogenous reconfiguration approach, which means that an entity termed as *Dynamic Reconfiguration Controller* (DRC) is responsible for managing every architectural change that occurs in the whole structure. DRC is an artificial architectural element that owns control over all elements of the software architecture and manages the architectural changes. DRC is added to the simulation to support the user in conducting architectural changes at runtime through the dynamic reconfiguration operators. Figure 2 presents the state machine of DRC. From the DEVS simulation model perspective, DRC is an atomic model, whose function is to execute the four dynamic reconfiguration operators while maintaining properties of the initial architecture configuration

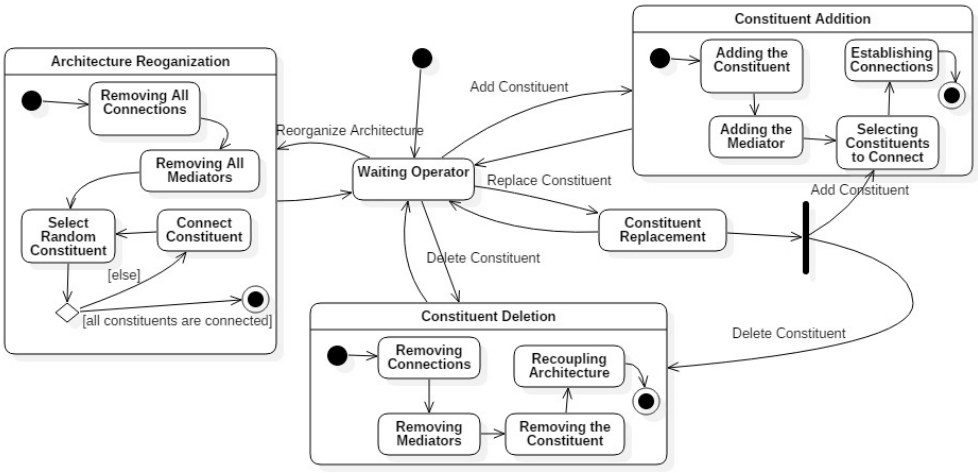

Fig. 2. State machine implemented in DEVS to materialize the four canonical reconfiguration operators performed by the Dynamic Reconfiguration Controller (Manzano et al. 2019).

To execute a reconfiguration operator, it is necessary to send a signal to the DRC with the operator to be performed and, if the operator is not a reorganization, the constituent in which the operator will be executed (in the addition the type of the constituent to be added) is also required.

2.2. Adding Support of Dynamic Reconfiguration through a Model Transformation

All SosADL elements are mapped to DEVS to create a functional simulation. Transformation rules automatically create the DRC from the concrete SosADL architecture and add it to the simulation model. This controller holds and makes available to the user the four dynamic reconfiguration operators.

The model transformation generates three main elements related to the dynamic architecture of the target simulation model, which are:

- **DRC**: As presented in Section 2.1, it consists of an atomic model that manages all changes in the simulation, as shown in Figure 2. For that, it manages connections

and mediators among constituents, so that the new arrangements remain consistent with the original architecture;

- **Identification flags**: In DEVS, there is no distinction between constituent systems and mediators, all of which are transformed into atomic models. However, this differentiation is necessary to maintain the functional architecture after running a dynamic reconfiguration operator. To artificially bring that to DEVS, we added two identification flags to the atomic models: one to check if the system is a mediator or not (a binary variable), and another that is the constituent type name, such as sensor, transmitter, or gateway; and
- **Connections of all constituents with the DRC in the coupled model**: This is necessary to enable the controller to communicate with all constituents and to remove some of them, if necessary.

3. Evaluation

Dynamic-SoS was evaluated through nine case studies, three of which were designed specifically for the evaluation of the approach, and the other six designed for the evaluation of other projects. Dynamic-SoS was able to simulate 395 constituents in a single simulation and successfully ran 113 hours of simulation (Table 1). Due to the large number of constituents and the long duration of the simulations, auxiliary mechanisms for evaluation were implemented, such as a log for human experts, and automatic procedures for measuring and delivering results regarding the delivery percentage of the behaviors.

Three case studies designed for Dynamic-SoS evaluation were conducted in three distinct domains (Manzano et al. 2019):

1. **Flood Monitoring SoS (FMSoS)**: a SoS with no central authority and being part of a smart city responsible for monitoring rivers that cross urban areas and notifying the population about potential floods that can quickly occur, causing huge damage and risk for the population. FMSoS is deployed along the river and its sensors are spread on the riverbank's edges and data are transmitted to a gateway. Additionally, drones fly over the river and return to their bases to recharge and communicate flood threat alerts. In parallel, people walking close to the river can also contribute by communicating water level increases through a crowdsensing mechanism supported by mobile apps;
2. **Space SoS**: a real system called Environmental Data Collection System (SBCDA, in Portuguese). Space SoS is operated and managed by the Brazilian National Institute for Space Research (INPE, in Portuguese), which provided us real data and know-how to precisely model the entire system. Space SoS is responsible to perform two concurrent missions (*environmental data collection* and *image capture*), and is composed of satellites, ground station, command and control center, and data collection platforms, which each of then can be managed by different institutions; and
3. **Smart Building**: a SoS that provides important services to their residents and visitors, such as energy savings, and light control by sensors, and fire alarms in case they happen. Dynamic-SoS approach was also evaluated using a Smart Building SoS (SBS), which is composed of three other SoS: fire system (composed of sensors, alarms, and sprinklers), lighting system (composed of sensors and lamps), and room (composed of sensors, air conditioners, and alarms). Moreover, SBS

has a Smart Building Control Unity (SBCU), which is responsible for managing constituent systems of the building.+

In this work, the **matter of interest** is *the simulation of SoS dynamic architectures*. We intended to observe whether Dynamic-SoS was well-succeeded for both: (i) automatically providing simulation models that support dynamic reconfigurations; and (ii) supporting the analysis of the impact of such reconfigurations on the SoS architecture, in particular, in regards to the sustainability of SoS emergent behaviors. In this way, the research question addressed in these case studies was *Are the architectural changes successful, giving rise to new SoS coalitions in a valid operational state?*

We executed three different evaluations, involving FMSoS, Smart Building, and Space SoS. For all of them, we applied the same set of architectural changes in the same order, and observed their impact on the data transmission rates (intimately aligned with the emergent behaviors provided by the SoS) expressed by the percentage of data that was successfully received in a gateway or that passed by a specific constituent in regards to the total number of data that was provided as stimuli to feed each simulation.

Figure 3 shows data collected from simulations. This data presents the percentage of collected data during the case studies (y axis) and performed operators (color, as specified in the legend). This data supports us to analyze the SoS availability and performance to fulfill its missions during its execution and predict how changes in the architecture of SoS will impact its performance, allowing to anticipate eventual problems.

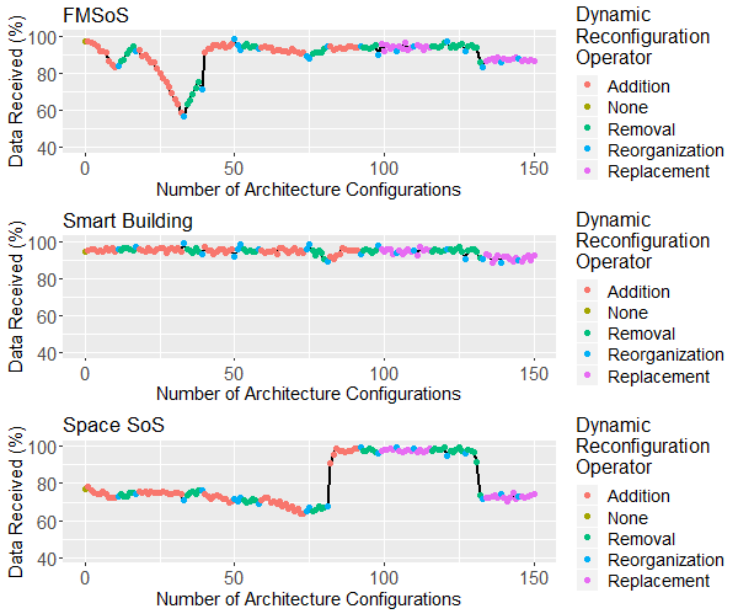

Fig. 3. Chart showing the rate of receipt of data throughout the case studies.

Additionally, we can analyze the impact of executing operators in the architecture. Figure 4 shows the impact of the addition operator on receiving data (y axis) and shows in which constituent type was performed the operator (point color). This chart allows us to analyze how the increase in the number of constituents affects data transmission. In all case studies, the addition of sensors/collectors decreased the data received, while the addition of constituents that receive these data increases the rate of data collection. This helps us to draw a trade-off on the acquisition of new constituents, making it possible,

for instance, to conclude that adding a second gateway in the FMSoS increases the data reception rate by 20%, which is the same gain that happens when adding a second satellite to the Space SoS. Then, after analyzing and triangulating results obtained from three

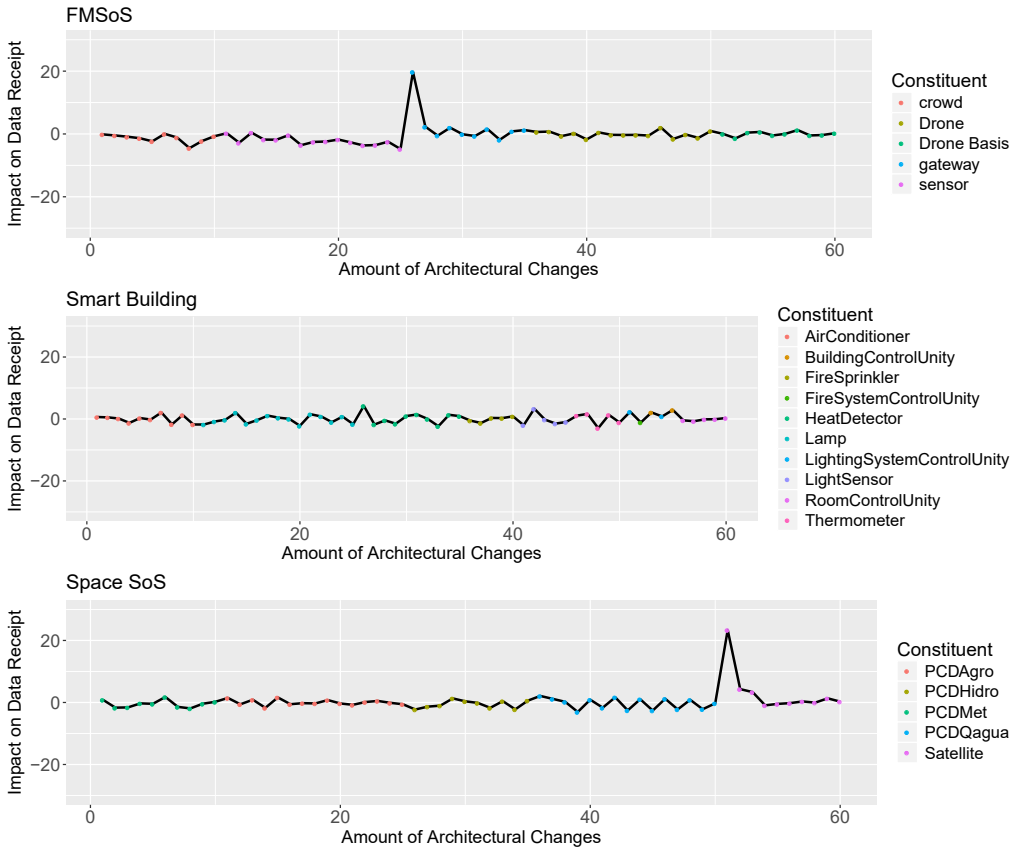

Fig. 4. Chart presenting the impact of the addition of a constituent in the simulation.

different independent studies and monitoring the logs file of the architectural state of the simulation, we can answer the raised research question for the case studies: *Are the architectural changes successful, giving rise to new SoS coalitions in a valid operational state?* The answer is Yes. In 100% of the cases, Dynamic-SoS was successful to support dynamic reconfigurations in SoS software architectures, leading the architecture to valid operational states in all obtained coalitions.

Besides the three case studies designed for the Dynamic-SoS evaluation, our approach supported the execution of other six cases, totaling nine case studies. Table 1 shows these studies, their domain, number of lines of code (LOC) of the models specified in SosADL, amount generated in DEVS by model transformation, maximum number of constituent reached in the case study, and total simulation time. With these results, we were able to evaluate that Dynamic-SoS was successful in generating more than 1 million LOC, and successfully generating simulations that ran more than 110 hours. Hence, we can answer our first research question: *How to provide a means to assure that the SoS operation is trustworthy, i.e., will it be maintained on-the-fly, despite the inherent SoS dynamic architecture?* The answer is that Dynamic-SoS can be used to maintain the simulation on-the-fly by using a reconfiguration controller to properly manage the reconfigurations in the architecture.

Tab. 1. Lines of code generated in each case study.

#	Study	Domain	SosADL (LOC)	DEVS (LOC)	Maximum Number of Constituents	Total simulation time (hours)
1	(Graciano Neto et al. 2018a)	FMSoS	701	8,563	70	6.33
2	(Manzano et al. 2019)	FMSoS	701	8,563	52	5.33
3	(Graciano Neto et al. 2017)	FMSoS	1,154	56,185	112	6.2
4	(Graciano Neto et al. 2018b)	FMSoS	660	3,332	89	2.6
5	(Manzano et al. 2019)	Smart Building	6,724	219,930	395	9.66
6	(Graciano Neto et al. 2018c)	Space SoS	640	11,195	4	0.72
7	(Graciano Neto et al. 2019)	Space SoS	2,399	128,884	311	27
8	(Manzano et al. 2019)	Space SoS	2,677	378,557	155	21.5
9	(Graciano Neto et al. 2019)	Space SoS	4,014	497,040	258	34
		Total	19,670	1,312,249	1,446	113.34

4. Contributions

Publications. This work resulted in four publications, **one in a journal, two in symposiums, and one awarded** as one of the three best works in an undergraduate student scientific works contest:

1. Manzano, W., Graciano Neto, V. V., and Nakagawa, E. Y. (2019).Dynamic-SoS: An Approach for the Simulation of Systems-of-Systems Dynamic Architectures. The Computer Journal, 63(5):709–731
2. Manzano, W., Graciano Neto, V. V., and Nakagawa, E. Y. (2018). Verificação Estatística de Modelos de Arquiteturas de Software de Sistemas-de-Sistemas. In 26o Simpósio Internacional de Iniciação Científica da USP (SIICUSP)
3. Manzano, W., Graciano Neto, V. V., and Nakagawa, E. Y. (2017). Simulação de Arquiteturas de Software de Sistemas-de-Sistemas com Suporte à Reconfiguração Dinâmica. In 25o Simpósio Internacional de Iniciação Científica da USP (SIICUSP)
4. W. Manzano, V. V. Graciano Neto, and E. Y. Nakagawa (2019). Simulating Systems-of-Systems Dynamic Architectures, 38º CTIC@CSBC, Belém, p. 1-10 (Also published in the Revista Eletrônica de Iniciação Científica)

In addition to these four self-authored important publications, other several publications were achieved using results provided by the approach developed by this student and having him as a co-author, including **two journal publications** (Graciano Neto et al. 2017; Graciano Neto et al. 2020), **two full papers in premier international conferences** (Graciano Neto et al. 2018c; Graciano Neto et al. 2018a), and also other important **four conferences and workshop papers**[4].

Impact. With regard to this work, important contributions were brought:

1. **Characterization of SoS dynamic architectures operators.** Providing dynamic architectures inherently depends on designing each possible change that can be performed on these architectures. Those changes should be well-defined and the final architecture should always deliver a valid operational state. Dynamic reconfiguration operators were previously established for single systems software architectures. However, when considering multiple interoperable systems present

[4] The list can be checked herein: `https://dblp.org/pers/hd/m/Manzano:Wallace`

in an SoS, there was a gap to be bridged. Dynamic-SoS builds a remarkable contribution on previous advances by providing a robust and canonical set of dynamic changes that can be adopted to represent any change that a SoS software architecture can suffer. We state this as a contribution that can be replicated in other contexts that require SoS dynamic architectures;

2. **Generation of dynamic models from static models.** Dynamic-SoS also provides a means to automatically derive SoS simulation models from static specification of SoS software architectures with a single initial coalition; and

3. **A process to include dynamics in SoS software architecture models.** By using a model transformation approach, we established a process that encompasses activities necessary to observe the dynamics of SoS via simulations, passing by the SoS software architecture specification step until reaching the assessment of different architectural configurations.

In addition, we extend Dynamic-SoS through a plugin to communicate with a statistical model checker to automatically perform a statistical verification of functional and architectural properties to automatically provide a level of trust in which the architecture is able to meet the properties. Besides, Dynamic-SoS can be expanded to cover other purposes of study, such as measuring other specific quality attributes, analyzing the threshold or limit of constituents that still maintain the SoS feasible, applying a set of random reconfigurations and studying the architecture behavior, and predicting the impact of specific constituents on the entire SoS architecture.

5. Final Remarks

SoS architectural design is a challenging task due to diverse architectural configurations that a SoS can assume at runtime and the complexity of assessing the impact of such changes still at design time. Dynamic-SoS contributes to this scenario by providing an infrastructure that allows SoS architects to predict, via simulations automatically generated at design-time, the SoS dynamic architectures, besides visualizing the SoS dynamics, and their impact on the SoS associated behavior. We conducted nine case studies, generating simulation models and analyzing the impact of changes on the functionalities provided by the modeled SoS. Results brought evidence that Dynamic-SoS is a feasible and effective approach to predict SoS behavior associated with the inherent dynamics associated with their architecture. Motivated by these results, our future work involves dealing with SoS dynamic architectures at runtime (i.e., after SoS deployment), advancing on *models@runtime* research branch. We intend to draw strategies for the synchronization between coalitions and SoS architectural specification, preserving SoS architectural consistency, avoiding degradation and assuring the quality of this class of systems that has been increasingly applied in several critical, complex application domains.

Acknowledgments

The authors thank São Paulo Research Foundation (FAPESP) (grants: 2014/02244-7, 2017/06195-9, 2017/17448-5, and 2018/21517-5) and National Council for Scientific and Technological Development - CNPq (grant: 312634/2018-8).

References

[Bogado et al. 2017] Bogado, V., Gonnet, S., and Leone, H. (2017). DEVS-based methodological framework for multi-quality attribute evaluation using software architectures. In *CLEI*, pages 1–10, Cordoba, Argentina.

[Cavalcante et al. 2015] Cavalcante, E., Batista, T. V., and Oquendo, F. (2015). Supporting dynamic software architectures: From architectural description to implementation. In *WICSA*, pages 31–40, Montreal, Canada.

[de França and Travassos 2016] de França, B. B. N. and Travassos, G. H. (2016). Experimentation with dynamic simulation models in software engineering: planning and reporting guidelines. *Empirical Software Engineering*, 21(3):1302–1345.

[Graciano Neto et al. 2020] Graciano Neto, V., Horita, F., dos Santos, R., Viana, D., Kassab, M., Manzano, W., and Nakagawa, E. (2020). S.o.b (save our budget) - a simulation-based method for prediction of acquisition costs of constituents of a system-of-systems. *iSys*, 13:6–35.

[Graciano Neto et al. 2017] Graciano Neto, V. V., Barros Paes, C. E., Garcés, L., Guessi, M., Manzano, W., Oquendo, F., and Nakagawa, E. Y. (2017). Stimuli-SoS: a model-based approach to derive stimuli generators for simulations of systems-of-systems software architectures. *Journal of the Brazilian Computer Society*, 23(1):1–22.

[Graciano Neto et al. 2018a] Graciano Neto, V. V., Garcés, L., Guessi, M., Paes, C., Manzano, W., Oquendo, F., and Nakagawa, E. Y. (2018a). ASAS: An approach to support simulation of smart systems. In *51st HICSS*, pages 5777–5786, Big Island, Hawaii, USA.

[Graciano Neto et al. 2018b] Graciano Neto, V. V., Manzano, W., Garcés, L., Guessi, M., Oliveira, B., Volpato, T., and Nakagawa, E. Y. (2018b). Back-SoS: Towards a model-based approach to address architectural drift in systems-of-systems. In *SAC*, pages 1–3, Pau, France. ACM.

[Graciano Neto et al. 2018c] Graciano Neto, V. V., Manzano, W., Rohling, A. J., Ferreira, M. G. V., Volpato, T., and Nakagawa, E. Y. (2018c). Externalizing patterns for simulations in software engineering of systems-of-systems. In *SAC*.

[Graciano Neto et al. 2019] Graciano Neto, V. V., Paes, C. E., Rohling, A. J., Manzano, W., and Nakagawa, E. Y. (2019). Modeling & simulation of software architectures of systems-of-systems : An industrial report on the brazilian space system. In *SpringSim*, pages 1–12. Accepted.

[Guessi et al. 2015] Guessi, M., Graciano Neto, V. V., Bianchi, T., Felizardo, K. R., Oquendo, F., and Nakagawa, E. Y. (2015). A systematic literature review on the description of software architectures for systems of systems. In *SAC*, pages 1433–1440, Salamanca, Spain.

[Guessi et al. 2016] Guessi, M., Oquendo, F., and Nakagawa, E. Y. (2016). Checking the architectural feasibility of systems-of-systems using formal descriptions. In *SoSE*, pages 1–6, Kongsberg, Norway.

[Heegaard and Schoitsch 2015] Heegaard, P. E. and Schoitsch, E. (2015). Introduction to the special theme - trustworthy systems of systems. *ERCIM News*, 2015(102):8–9.

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

[Manzano et al. 2019] Manzano, W., Graciano Neto, V. V., and Nakagawa, E. Y. (2019). Dynamic-SoS: An Approach for the Simulation of Systems-of-Systems Dynamic Architectures. *The Computer Journal*, 63(5):709–731.