A Systematic Mapping on Executable Models for the Architectural Design of Systems-of-Systems

Bruno G. A. Lebtag[2], Paulo Gabriel Teixeira[2], Valdemar Vicente Graciano Neto[1]

[1]Informatics Institute – Federal University of Goiás (UFG)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

[2]SiDi – Campinas – SP – Brazil

{b.lebtag,p.teixeira}@sidi.org.br, valdemarneto@ufg.br

Abstract. *Systems-of-Systems (SoS) are complex systems composed of managerially and operationally independent constituent systems (CS). Smart cities are examples of SoS. These types of systems impose challenges to the traditional software architecture design, such as describing heterogeneous CS that are constantly evolving and identifying emergent behaviors from the interactions of those CS. Executable models (ExM) have been envisioned as a possible solution to deal with the challenges raised by SoS architectural design. However, to the best of our knowledge, no systematic mapping study (SMS) exists that investigate current state of art in the research area of SoS architectual design using ExM. Therefore, the main goal of this study is to synthesize evidence of that research area, observing research trends and identifying possible research gaps yet to be explored. Results reveal that (i) the use of model transformation from a static SoS domain-specific language to a ExM is intensely explored and (ii) ExM were used mainly to evaluate and synthesize SoS architectures but also to observe emergent behaviors and to measure quality attributes.*

1. Introduction

Smart cities are instances of System of Systems (SoS), i.e, alliances of multiple independent systems that work together to achieve complex behaviors [Oquendo et al. 2016]. The ascension of smart cities brings important concerns about safety, since failures can cause injuries, deaths or even environmental damage. Moreover, the complexity, large scale and inherent dynamic architecture of those systems makes it difficult to manage the entire system and guarantee that the changes will not impact on the functionalities being provided.

Executable models (ExM), i.e. models written in a language that offers execution semantics to express the behavior of the model [Hojaji et al. 2019], in which we can include simulation models and models@runtime, can assist engineers to manage those issues at design-time. ExM offer the ability to (i) benchmark the diverse arrangements a complex system can assume, (ii) predict system structure and behavior, despite its inherent dynamics and adaptivity and (iii) predict the consequences of architectural changes that can take place on-the-fly over the structure and behavior of such system of interest, still at design-time. These advantages match the requirements imposed by SoS. Software architects can use simulation models to study the complexity of the problem by exploring different scenarios and observing them visually. On the other hand, when the volume

of entities to be simulated hinders the possibility to have a meaningful visual representation, software architects can still rely on the execution trace of the simulation models [Hojaji et al. 2019].

However, to the best of our knowledge, no systematic mapping study (SMS) exists that investigate current state of art in the use of ExM to architectural design SoS. Therefore, to address this, we conducted a SMS on ExM in the context of SoS architectures. The main contribution of this paper is a synthesis of current research area and state of the art in the architectural design of SoS using ExM. We extracted data from 35 out of 196 pre-selected studies. We were able to identify historical tendencies of investigations strategies in the research area such as the progressive use of automatic synthesis of executable models as well as missing gaps yet to be explored by researchers such as the direct use of executable models (e.g. xUML) as the final product. The paper is structured as follows: In Section 2, we describe the SMS protocol. In Section 3, we report the obtained results and present our findings. In Section 4, we discuss the obtained results and its implications for the research area besides related work and threats to validity. Lastly, in Section 5, we conclude the paper with final remarks.

2. Research Protocol

The goal of this SMS is *to map current use of ExM for SoS architecture design as reported by the literature*. This goal was then refined into the following research questions, each one with a correspondent aim:

RQ1 — What research treads can be observed over the years? We want to understand how the use of ExM by researchers evolved over the years and to identify publication patterns and venues.

RQ2 — How ExM have been used for SoS architecture design? We want to understand how ExM was explored by researchers to design SoS in the architecture design life cycle, the results obtained by using ExM, in what type of SoS were ExM explored and whether there was any difference in treatment or obtained results. RQ2 was splited in two sub-questions, as follows:

RQ2.1 — What are the goals behind the use of ExM for SoS architectural design? ExM have known advantages over static notations. We want to identify the main goals researchers envisioned when they decided to use ExM for SoS architectural design.

RQ2.2 — What notations have been used? We want to produce a list of the most used notations reported in the literature for SoS architectural description and their characteristics. Because it is common to use model transformation to achieve executability, we are also interested in what static notations were used and to what ExM notations were they converted.

Our search and selection process was composed of five steps. We (i) developed the search string (listing 1) based on the experience of the authors and system engineering body of knowledge[SEBoK 2017], (ii) performed an automatic search without snowballing in the five largest scientific databases in system and software engineering, as suggested in Kitchenham and Charters [Kitchenham and Charters 2007]: ACM Digital Library, Engineering Village, IEEE Digital Library, Web of Science and Scopus, (iii)

removed unrelated or duplicated studies, in which we removed 35 studies, (iv) filtered the obtained results using inclusion and exclusion criteria and performed by authors, and lastly (v) performed an adaptive reading depth [Petersen et al. 2008] of the studies and removed studies that did not satisfy the inclusion criteria as well as we extracted the results to answer our RQs.

```
("architecture design" OR "architectural design" OR "
    architecture view" OR "architectural view" OR "
    architecture description" OR "architectural description"
     OR "architecture documentation" OR "architectural
    documentation" OR "architecture specification" OR "
    architectural specification" OR "architecture model" OR
    "architectural model") AND ("dynamic model" OR "dynamic
    system" OR "dynamic architecture" OR "simulation model"
   OR "simulation system" OR "simulation" OR "executable
    model" OR "executable system" OR "executable
    architecture" OR "executable runtime" OR "runtime model"
     OR "models@runtime" OR "models@run.time" OR "model
    execution" OR "executable UML" OR "GEMOC model" OR "
    digital twin") AND ("systems of systems" OR "system of
    system" OR "systems-of-systems" OR "system-of-system" OR
     "systems-of-system")
```

Listing 1. Search string used in the studies. Adapted to each search base.

The following *inclusion criteria* were defined for this SMS: (IC1) This study addresses the use of ExM for SoS architecture design, (IC2) This study addresses the proposal of a new ExM technology for SoS architectural design, (IC3) This study addresses the proposal of a new ExM notation for SoS architectural design, (IC4) This study addresses the interoperability of different ExM tools for SoS architectural design. And, the following *exclusion criteria* was used: (EC1) This study is not in English, (EC2) This study is not a primary study, i.e., it is a literature review (secondary study) or a non-peer-reviewed document, (EC3) This study do not address systems of systems, (EC4) This study do not address executable models, (EC5) This study do not address architectural design, (EC6) The study was not available to be accessed by the researchers, (EC7) The study is an old version of another study already considered.

3. Reporting

The included studies are the following: (s1) [Teixeira et al. 2020], (s2) [Binder et al. 2019], (s3) [Manzano et al. 2019], (s4) [Beery et al. 2019], (s5) [Graciano Neto et al. 2018b], (s6) [Zhang et al. 2018], (s7) [Graciano Neto et al. 2018a], (s8) [Zhuang et al. 2018], (s9) [Esmaeilzadeh et al. 2018], (s10) [Dahmann et al. 2017], (s11) [Graciano Neto et al. 2017], (s12) [Silva et al. 2017], (s13) [Hachem et al. 2016], (s14) [Oquendo et al. 2016], (s15) [Li et al. 2013], (s16) [Yousefi and Levis 2016], (s17) [Arnold et al. 2016], (s18) [Spichkova et al. 2015], (s19) [Li et al. 2016b], (s20) [Wang et al. 2015], (s21) [Cavalcante 2015], (s22) [Wätzoldt and Giese 2015], (s23) [Perišić et al. 2015], (s24) [Hu et al. 2014], (s25) [Hsu et al. 2014], (s26) [Rieckmann et al. 2013], (s27) [Ge et al. 2013], (s28) [Li et al. 2013], (s29)

[Ludwig et al. 2011], (s30) [Wang and Dagli 2011], (s31) [Xiong et al. 2010], (s32) [Garcia and Tolk 2010], (s33) [Sindiy et al. 2009], (s34) [Robbins 2009], (s35) [Rao et al. 2008]. We answer the raised research questions, as follows.

RQ1 — *What research treads can be observed over the years?*

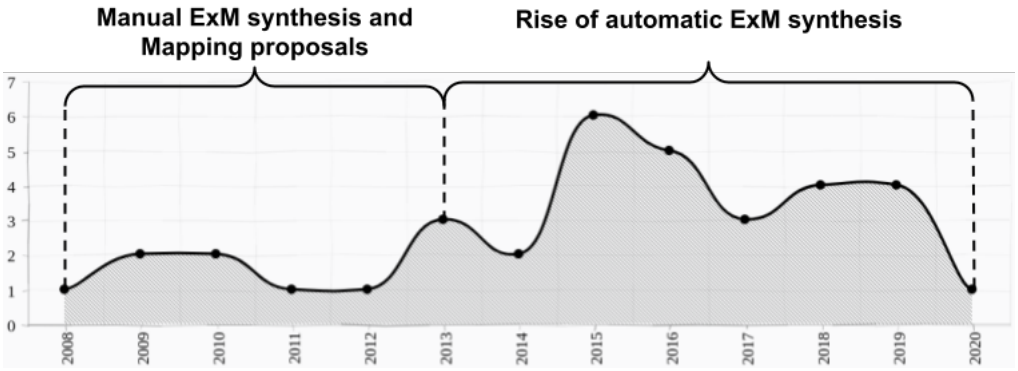

Figure 1. Distribution of studies per year.

Figure 1 presents the distribution of included studies over the years. The first published study that reports the use of ExM for SoS architecture design was Rao et al. [Rao et al. 2008], in 2008. They proposed a mapping strategy between SysML and Colored Petri Nets (CPN) and performed an architectural evaluation. From 2008 to 2013, we identified five others consecutively mapping proposals between different static notations to ExM notations where the ExM is simulated to perform architectural analysis or evaluation [Garcia and Tolk 2010, Li et al. 2013, Robbins 2009, Wang and Dagli 2011, Xiong et al. 2010]. Then, since 2013, we observed the rise of new proposals that offered automatic model transformations. Studies could then be separated in two categories. In the first category, the authors presented simple model transformation solutions that convert from one static notation to one executable notation. In the second category, the studies offer more robust solutions, usually integrated into a development environment (IDE) covering different architecture life cycle activities within the same toolset.

Interestingly, even after the rise of automatic transformation approaches, it is still possible to identify studies that manually synthesize the architecture proposal into ExM [Beery et al. 2019, Esmaeilzadeh et al. 2018, Hsu et al. 2014, Teixeira et al. 2020, Wang et al. 2015, Yousefi and Levis 2016]. No convergence in the community towards a common technology or tool was observed. In fact, different studies targeted different approaches to use ExM for analysing and evaluating the SoS architecture. Finally, the newest analysed study adopted a different approach apart from the remaining included studies. Teixeira et al. [Teixeira et al. 2020] focused on designing constituent systems (CSs) to be part of a SoS. The study differs from the other studies because, while the other studies adopted a top-down approach for SoS architecture design (SoS to constituents), that study employed a bottom-up approach for SoS architecture design (constituents to SoS). The study also shifts the focus to use ExM for developing the constituent while the majority of the other studies focuses on the development of the SoS in its entirety.

Finding 1: *In the beginning of the research area, studies were composed of mapping strategies between static notations to ExM notations, in which the transformation itself*

was executed manually. Thereafter, studies proposing automatic model transformation emerged. However, it is still possible to find studies conducting manual transformations.

RQ2.1 — *What are the goals behind the use of ExM for SoS architectural design?*

Figure 2 presented the goals we identified in the studies. *Architectural evaluation and synthesis* was explored in (24/35) studies [Arnold et al. 2016, Beery et al. 2019, Dahmann et al. 2017, Esmaeilzadeh et al. 2018, Garcia and Tolk 2010, Graciano Neto et al. 2018a, Graciano Neto et al. 2018b, Hu et al. 2014, Li et al. 2013, Li et al. 2016a, Ludwig et al. 2011, Manzano et al. 2019, Oquendo et al. 2016, Rao et al. 2008, Rieckmann et al. 2013, Robbins 2009, Sindiy et al. 2009, Teixeira et al. 2020, Wang et al. 2015, Wang and Dagli 2011, Xiong et al. 2010, Yousefi and Levis 2016, Zhang et al. 2018, Zhuang et al. 2018]. Several studies presented new methods to perform architectural design using ExM via synthesis of the architecture using model transformation from static notations such as UML, SysML SoaML and DoDAF to ExM notations in an automatically or manually approach.

Another important goal investigated in the studies was *emergent behavior analysis* (17/35) [Arnold et al. 2016, Binder et al. 2019, Esmaeilzadeh et al. 2018, Garcia and Tolk 2010, Ge et al. 2013, Graciano Neto et al. 2017, Hachem et al. 2016, Hsu et al. 2014, Hu et al. 2014, Manzano et al. 2019, Oquendo et al. 2016, Perišić et al. 2015, Robbins 2009, Silva et al. 2017, Sindiy et al. 2009, Wang et al. 2015, Wang and Dagli 2011]. *Quality Attribute Analysis* was also highly explored (10/35) [Hu et al. 2014, Perišić et al. 2015, Rieckmann et al. 2013, Sindiy et al. 2009, Spichkova et al. 2015, Wang et al. 2015, Wang and Dagli 2011, Xiong et al. 2010, Zhang et al. 2018, Zhuang et al. 2018].

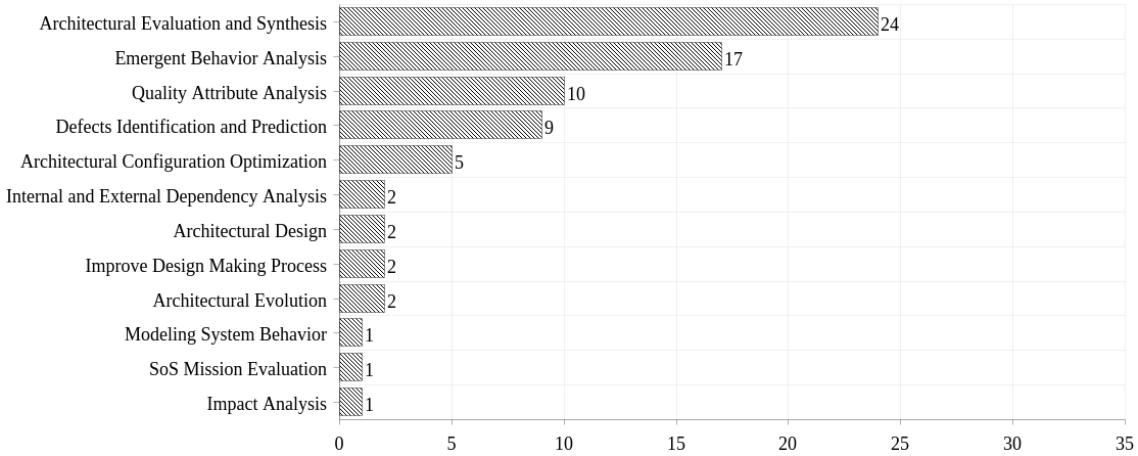

Figure 2. Goals reported by studies.

Finding 2: *Studies explored different strategies to synthesize the architecture and evaluate it using ExM. Quality Attributes (QA) analysis was also relevant for researchers. ExM were used: to (i) generate values to manually measure QA, (ii) measure QA automatically, (iii) run the simulation and observe its behavior and identify possible problems, and (iv) allow the architecture to evolve by harvesting ExM natural ability to evolve.*

RQ2.2 — *What notations have been used?*

For SoS architectural design, the main adopted strategy is to design the system using static notations and then apply model transformation to another ExM notation or directly defining the operational semantics for the static notation elements. Because SoS is particularly relevant to the military domain, *DoDAF* was the most adopted static notation for architectural design (5/35) [Garcia and Tolk 2010, Li et al. 2016a, Robbins 2009, Zhang et al. 2018, Zhuang et al. 2018]. DoDAF is a architectural framework used by the US DoD to enable the development of architectures by facilitating the management of decision making process and organizing the sharing of information. DoDAF offers different views that cover different aspects of the architecture and it is the *de facto* framework used by the US military. The second most used static notation for architectural design was SysML which is a UML2.0 profile for modeling and engineering system applications (4/35) [Dahmann et al. 2017, Hu et al. 2014, Rao et al. 2008, Wang and Dagli 2011]. DoDAF and SysML are general purpose modeling languages that focus in system architecture in a general sense. SoSADL, on the other hand, is an ADL developed specifically for SoS architectural design. SosADL was in the third position (3/35) [Graciano Neto et al. 2017, Manzano et al. 2019, Silva et al. 2017].

Finding 3: *Model transformation was the main strategy employed by researchers. The goal consisted of using a static domain specific notation such as DoDAF, SysML or SosADL to an ExM that is capable of being executable.*

In this study, DEVS was the most used ExM notation (9/35) [Garcia and Tolk 2010, Graciano Neto et al. 2018a, Graciano Neto et al. 2017, Graciano Neto et al. 2018b, Hu et al. 2014, Manzano et al. 2019, Oquendo et al. 2016, Silva et al. 2017, Teixeira et al. 2020]. DEVS is a state machine based formalism for analysing systems. CPN ML was the second most used ExM formalism (5/35) [Ge et al. 2013, Rao et al. 2008, Wang et al. 2015, Wang and Dagli 2011, Yousefi and Levis 2016]. CPN ML is a extension to petri nets that allows the distinction of tokens [Jensen 1997]. CPN ML was one of the first proposed ExM formalism to be used together with C4ISR (later renamed and remolded to DoDAF) via model transformation for architectural design [Levis and Wagenhals 2000]. Both DEVS and CPN ML are discrete event based formalism with strong mathematical definition which justifies the fact the are the most used modeling languages. Beyond that, Agent-based simulation (ABS) was also explored. ExtendSim notation was used in (2/35) studies [Hsu et al. 2014, Xiong et al. 2010]. ExtendSim is a simulator that offers discrete event simulation (such as DEVS) and ABS. It has a visual notation where the user can specify the architecture using semantic blocks that represent action. Other studies also explored ABS but it did not informed its formalism (2/35) [Esmaeilzadeh et al. 2018, Sindiy et al. 2009]. In fact, (14/35) studies did not informed the used notation [Beery et al. 2019, Cavalcante 2015, Esmaeilzadeh et al. 2018, Sindiy et al. 2009, Spichkova et al. 2015].

Finding 4: *DEVS was the most used ExM notation followed by CPN ML. Both notations represent 40% of the studies (14/35). Therefore, we observed a tendency towards discrete event based ExM for SoS architecture design.*

4. Discussion

In this SMS, we were able to observe the evolution of the approaches and usage of ExM for SoS architecture design (Finding 1). The research area, in the beginning, explored manual approaches for dealing with SoS. Due to their complexity and number of constituents, this approach soon became unmanageable. Therefore, new approaches using automatic transformation were created. More recently, studies started offering complete solutions that cover more software architecture life cycle activities. This is important for the research area because many studies were explored in critical domains, such as military.

From SoS and software architecture perspective, researchers explored the SoS architectural design focusing on the SoS architectural analysis and evaluation (Finding 2). All SoS characteristics were explored. The researchers aimed to identify unexpected emergent behaviors that can impair the SoS architecture and measure the QA of the explored SoS. The execution and achievement of missions, which are the main goal of a SoS, did not received direct attention. This was the main focus of only one study [Silva et al. 2017], which worked to refine mission models into architectural descriptions.

Only one study that offered code generation [Li et al. 2016a]. No studies exploring the use of ExM for architectural implementation that could be directly interpreted without the need of model transformation such as executable UML (xUML) were identified [OMG Executable UML 2018] (Finding 3). xUML allows software architects to use UML diagrams to represent the structures of the system being developed and by using action languages such as Alf [OMG Executable UML 2017] to add instructions to those structures to be executed when activated. We can highlight that such approaches has low adoption in the industry and thus could explain the lack of interest from researchers to explore them. We observed that Researchers, instead preferred to perform model transformation between some static notation such as DoDAF, UML or SysML to ExM formalism such as DEVS, Petri Nets and ExtendSim Models (Finding 3). Authors did not provide reasons for the why but what can be inferred from the studies we could point to the fact that those static notations are closer related to the explored domains such as DoDAF is for military domain. Graciano Neto et al. [Graciano Neto et al. 2014] also found similar results in their systematic literature review. The authors identified 83.3% (10/12) studies using model transformation from different notations such as UML, DoDAF and SysML, but they did not find any study using xUML or Alf.

Lastly, ExM supported researchers mainly via modeling the SoS and simulating it to observe its behaviors and properties (Finding 3 and 4). This study identified no convergence on techniques or tools for SoS architectural design (Finding 4). Discrete event, based on DEVS and CPN formalisms, was the most explored simulation technique but agent-based technique was also explored, mostly to simulate component's interactions of systems. ExM was used as the enabler for performing architectural analysis and evaluation but not as the final delivered product. We did not observe any study using for instance xUML, which is capable of executing UML, being explored. Thus, this potential use of ExM is an open research area.

Related Work. We could find other literature reviews related to the use of ExM and/or SoS, such as Ciccozzi et al. [Ciccozzi et al. 2019], Hojaji et al. [Hojaji et al. 2019], and Guessi et al. [Guessi et al. 2015]. To the best of our knowledge, there is no prior SMS

that investigated the use of ExM for SoS architectural design, as we perform herein.

Threats to validity. Concerning included studies, we could have missed relevant studies. To mitigate this threat, an automatic search was conducted in five of the main scientific databases and and indexing systems. Regarding how the study was structured and conducted, potential threats were mitigated by (i) rigorously defining and following the protocol of this study, and (ii) defining and using a data extraction form which was developed to answer the research questions of this study. Regarding the validity of the synthesis, potential threats were mitigated by employing descriptive statistics calculated by a statistic software. Potential threats to conclusion validity were mitigated by applying well-accepted systematic processes throughout our study and documenting all of them in the research protocol. Thus, other researchers can replicate this study accordingly. The data extraction phase could also have been another source of threats to the conclusion validity of this study. The bias was mitigated by (i) performing a thoroughly read while attempting to make as few assumption as possible and (ii) having the data extraction process conducted by one researcher but validated by other researchers.

5. Conclusions

The main contribution of this paper was to report results of a Systematic Mapping Study (SMS) about the adoption of executable models (ExM) in the architectural design of Systems-of-Systems (SoS). We analysed 35 studies from a total 196 potential studies. From the collected data and data synthesis, we observed that researchers are exploring ExM: (i) to analyse, evaluate and synthesize the SoS architecture using ExM (ii) to be able to measure quality attributes and analysis different architectural configurations, (iii) together with model transformation to convert static notations such as DoDAF, UML and SysML to ExM notations such as DEVS, Petri Nets and ExtendSim models and (iv) emergent behavior and evolutionary development mostly explored by researchers. Researchers did not used ExM such as xUML with Alf but instead preferred to use model transformation and more formal ExM notations. We also did not observe ExM to explore the modeling of SoS missions which is one of the most important goals of the SoS. Therefore, we claim that there are many open opportunities to be explored in the SoS architectural design using ExM. In this sense we can enumerate few gaps that can be explored as future work, such as: (i) the need for exploring mission modeling using ExM, (ii) consolidate current practices in the use of ExM for SoS architectural design, and (iii) consolidation of current tooling in the use of ExM architectural design.

Acknowledgements This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This work would not have been possible without the support by SiDi Institute of Research and Development.

References

Arnold, A., Baleani, M., Ferrari, A., Marazza, M., Senni, V., Legay, A., Quilbeuf, J., and Etzien, C. (2016). An application of smc to continuous validation of heterogeneous systems. SIMUTOOLS'16, page 76–85.

Beery, P., Enloe, M., Kummer, G., Paulo, E., Kelly, E., Watson, S., Smith, S., Kummer, K., Corbett, L., and Jankowski, B. (2019). Command and control for distributed lethality.

Binder, C., Gross, J.-A., Neureiter, C., and Lastro, G. (2019). Investigating emergent behavior caused by electric vehicles in the smart grid using co-simulation. pages 230–235.

Cavalcante, E. (2015). On the architecture-driven development of software-intensive systems-of-systems. ICSE '15, page 899–902.

Ciccozzi, F., Malavolta, I., and Selic, B. (2019). Execution of uml models: a systematic review of research and practice. 18(3):2313–2360.

Dahmann, J., Markina-Khusid, A., Doren, A., Wheeler, T., Cotter, M., and Kelley, M. (2017). Sysml executable systems of system architecture definition: A working example.

Esmaeilzadeh, E., Grenn, M., and Roberts, B. (2018). An agent-based model for improved system of systems decision making in air transportation. Systems Engineering, 22(1):20–42.

Garcia, J. and Tolk, A. (2010). Adding executable context to executable architectures: Shifting towards a knowledge-based validation paradigm for system-of-systems architectures. SCSC '10, page 593–600, San Diego, CA, USA.

Ge, B., Hipel, K., Yang, K., and Chen, Y. (2013). A novel executable modeling approach for system-of-systems architecture. 8(1):4–13.

Graciano Neto, V. V., Barros Paes, C. E., Garces, 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. JBCS, 23(1).

Graciano Neto, V. V., Guessi, M., Oliveira, L. B. R., Oquendo, F., and Nakagawa, E. Y. (2014). Investigating the model-driven development for systems-of-systems. ECSAW '14. ACM.

Graciano Neto, V. V., Manzano, W., Garcés, L., Guessi, M., Oliveira, B., Volpato, T., and Nakagawa, E. Y. (2018a). Back-sos: Towards a model-based approach to address architectural drift in systems-of-systems. SAC '18, page 1461–1463.

Graciano Neto, V. V., Manzano, W., Kassab, M., and Nakagawa, E. Y. (2018b). Model-based engineering & simulation of software-intensive systems-of-systems: Experience report and lessons learned. ECSA '18.

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 30th ACM SAC, pages 1433–1440.

Hachem, J. E., Chiprianov, V., Babar, A., and Aniorte, P. (2016). Towards methodological support for secure architectures of software-intensive systems-of-systems. SiSoS@ECSA '16.

Hojaji, F., Mayerhofer, T., Zamani, B., Hamou-Lhadj, A., and Bousse, E. (2019). Model execution tracing: a systematic mapping study. SoSyM, 18(6):3461–3485.

Hsu, J., Price, M., Clymer, J., Garcia Jr., J., and Gonzalez, E. (2014). Agent-based modeling the emergent behavior of a system-of-systems. volume 3, pages 1581–1590.

Hu, J., Huang, L., Chang, X., and Cao, B. (2014). A model driven service engineering approach to system of systems. pages 136–145.

Jensen, K. (1997). Coloured Petri Nets. Springer Berlin Heidelberg.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. 2:1–65.

Levis, A. H. and Wagenhals, L. W. (2000). C4isr architectures: I. developing a process for c4isr architecture design. Systems Engineering, 3(4):225–247.

Li, X., Wang, W., Shu, Z., Zhu, N., He, H., and Liao, T. (2016a). A system-of-systems architecture-driven modeling method for combat system effectiveness simulation.

Li, Z., Zhu, Y., and Yang, F. (2016b). Sos architecture alternatives tradespace modeling and computable experimentation: A framework with system engineering thinking. pages 677–681.

Li, Z.-F., Qin, D.-L., Yuan, H., Yang, F., and Zhu, Y.-F. (2013). A method of wesos capability assessment based on dm2 and abs. volume 3, pages 1–6.

Ludwig, M., Farcet, N., Babau, J.-P., and Champeau, J. (2011). Integrating design and runtime variability support into a system adl. 6698 LNCS:270–281.

Manzano, W., Graciano Neto, V., and Nakagawa, E. (2019). Dynamic-sos: An approach for the simulation of systems-of-systems dynamic architectures. Computer Journal, 63(5):709–731.

OMG Executable UML (2017). Action language for alf. Standard, Object Management Group, Massachusetts, USA.

OMG Executable UML (2018). Semantics of fuml. Standard, Object Management Group, Massachusetts, USA.

Oquendo, F., Buisson, J., Leroux, E., Moguérou, G., and Quilbeuf, J. (2016). The sosadl studio: An architecture development environment for software-intensive systems-of-systems. SiSoS@ECSA '16.

Perišić, A., Lazic, M., Perišić, B., and Obradovic, R. (2015). A smart house environment - the system of systems approach to model driven simulation of building (house) attributes. pages 56–59.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. EASE'08, page 68–77.

Rao, M., Ramakrishnan, S., and Dagli, C. (2008). Modeling and simulation of net centric system of systems using systems modeling language and colored petri-nets: A demonstration using the global earth observation system of systems. Systems Engineering, 11(3):203–220.

Rieckmann, M., Fraser, D., Chiprianov, V., Szabo, C., and Falkner, K. (2013). Demonstration of model-driven performance prediction of distributed real-time embedded systems of systems. ECSAW '14.

Robbins, W. (2009). Achieving dodaf-driven simulations through executable architectures. SpringSim '09.

SEBoK (2017). The systems engineering body of knowledge (sebok), version 1.8.

Silva, E., Cavalcante, E., and Batista, T. (2017). Refining missions to architectures in software-intensive systems-of-systems. pages 2–8.

Sindiy, O., DeLaurentis, D., and Stein, W. (2009). An agent-based dynamic model for analysis of distributed space exploration architectures. JAS, 57(3):579–606.

Spichkova, M., Liu, H., and Schmidt, H. (2015). Towards quality-oriented architecture: Integration in a global context. ECSAW '15.

Teixeira, P. G., Lebtag, B. G. A., d. Santos, R. P., Fernandes, J., Mohsin, A., Kassab, M., and Neto, V. V. G. (2020). Constituent system design: A software architecture approach. ICSA-C '20, pages 218–225.

Wang, R., Agarwal, S., and Dagli, C. (2015). Opm & color petri nets based executable system of systems architecting: A building block in fila-sos. pages 554–561.

Wang, R. and Dagli, C. (2011). Executable system architecting using systems modeling language in conjunction with colored petri nets in a model-driven systems development process. Systems Engineering, 14(4):383–409.

Wätzoldt, S. and Giese, H. (2015). Modeling collaborations in adaptive systems of systems. ECSAW '15.

Xiong, J., Ge, B.-F., Zhang, X.-K., Yang, K.-W., and Chen, Y.-W. (2010). Evaluation method of system-of-systems architecture using knowledge-based executable model. pages 141–147.

Yousefi, B. and Levis, A. (2016). Architecture-based simulation for system evaluation. volume 48, pages 39–46.

Zhang, M., Chen, H., Zhang, X., Luo, A., and Liu, J. (2018). Functionality evaluation of system of systems architecture based on extended influence diagrams. JSEE, 29(3):510–518.

Zhuang, Z., Lin-lin, L., and Hong-feng, Y. (2018). Architecture modeling of new c2 system of joint anti-ship combat. ICACS '18, page 248–252.