

A Systematic Identification of Formal and Semi-formal Languages and Techniques for Software-intensive Systems-of-Systems Requirements Modeling

Cristiane Aparecida Lana, Milena Guessi, Pablo Oliveira Antonino, Dieter Rombach, and Elisa Yumi Nakagawa

Abstract—Software-intensive Systems-of-Systems (SoS) refer to an arrangement of managerially and operationally independent systems (i.e., constituent systems), which work collaboratively towards the achievement of global missions. Because some SoS are developed for critical domains, such as healthcare and transportation, there is an increasing need to attain higher quality levels, which often justifies additional costs that can be incurred by adopting formal and semi-formal approaches (i.e., languages and techniques) for modeling requirements. Various approaches have been employed, but a detailed landscape is still missing, and it is not well known whether they are appropriate for addressing the inherent characteristics of SoS. The main contribution of this article is to present this landscape by reporting on the state of the art in SoS requirements modeling. This landscape was built by means of a systematic mapping and shows formal and semi-formal approaches grouped from model-based to property-oriented ones. Most of them have been tested in safety-critical domains, where formal approaches such as finite state machines are aimed at critical system parts, while semi-formal approaches (e.g., UML and i*) address non-critical parts. Although formal and semi-formal modeling is an essential activity, the quality of SoS requirements does not rely solely on which formalism is used, but also on the availability of supporting tools/mechanisms that enable, for instance, requirements verification along the SoS lifecycle.



1 Introduction

DU E to the accelerated development of industry and society, several independent software-intensive systems are now able to exchange information and interoperate with each other, resulting in more complex systems, which are called Systems-of-Systems (SoS¹) [1], [2], [3], [4], [5]. These constituent systems are often developed by different companies and rely on different platforms and technologies. The combined work of the constituents allows the SoS to perform complex functions that could not be delivered otherwise [1], [6]. In this sense, two of the distinguishing characteristics presented by SoS are their evolutionary development and their emergent behavior [1], [4], [7], [8]. *Evolutionary development* refers

to the capacity of an SoS to evolve in response to changes in its environment, its constituent systems, or its missions, functions, and purposes. For instance, an SoS must absorb the constituents' changes, which may affect its mission to resume its proper functioning. *Emergent behavior* refers to new functions that cannot be realized by any constituent system separately. In fact, these behaviors (or functionalities) can only be realized by the interactions among constituents over time [9]. Emergent behaviors may originate in constituents and trigger new behaviors at the SoS level and vice-versa. Examples of SoS can be found in diverse application domains, such as military [10], aerospace [11], transportation [12], and health care [13], and are becoming a trend for future systems.

In parallel, Requirements Engineering (RE) is recognized as a fundamental activity of successful software development processes and comprises modeling², verification, and validation (V&V)³ of requirements. For researchers and industry professionals, software projects are extremely vulnerable when RE is conducted poorly [15]. In general, 70% of software projects fail due to low-quality requirements, whose rework cost exceed \$45 billion annually [16]. The cost for correcting errors⁴ originating in requirements and persisted throughout different development phases can scale up to 100 times of

• C. A. Lana is with the Institute of Mathematics and Computer Sciences (ICMC), University of São Paulo (USP), São Carlos 13566-590, Brazil, with the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE), 67663 Kaiserslautern, Germany, and also with Technische Universität Kaiserslautern (TUK), 67663 Kaiserslautern, Germany (cristiane.lana@usp.br; lana@rhrk.uni-kl.de)

M. Guessi is with the Institute of Mathematics and Computer Sciences (ICMC), University of São Paulo (USP), São Carlos 13566-590, Brazil (milena@icmc.usp.br)

P. O. Antonino is with the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE), 67663 Kaiserslautern, Germany (pablo.antonino@iese.fraunhofer.de)

Dr. D. Rombach is with the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE), 67663 Kaiserslautern, Germany and also with Technische Universität Kaiserslautern (TUK), 67663 Kaiserslautern, Germany (dieter.rombach@iese.fraunhofer.de)

E. Y. Nakagawa is with the Institute of Mathematics and Computer Sciences (ICMC), University of São Paulo (USP), São Carlos 13566-590, Brazil (elisa@icmc.usp.br)

This version was submitted to IEEE System Journal on 24 February 2018

1. For the sake of simplicity, the acronym SoS is interchangeably used to express singular and plural.

2. The terms modeling and specification are used interchangeably in this article.

3. We consider V&V of requirements as defined in ISO/IEC/IEEE 24765:2017 [14]. **Validation**: confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled. **Verification**: the process of ensuring that the software requirements specification complies with the system requirements, conforms to document standards of the requirements phase, and is an adequate basis for the architectural (preliminary) design phase

4. We consider an error as a human action that produces an inconsistent requirements document.

their original costs [17], [18]. To address this risk, RE adopts informal, semi-formal, and formal techniques, methods, and languages (notations) for requirements modeling and V&V [19]. Informal notations are usually more expressive and flexible but they also rely on human expertise [20]. On the other hand, semi-formal languages⁵, e.g., i*, Unified Modeling Language (UML), and Systems Modeling Language⁶ (SysML), provide a defined syntax but lack complete semantics to support communication among the stakeholders [23]. Alternatively, formal languages, such as the Vienna Development Method (VDM) and Larch, provide both a well-defined syntax and semantics, being a set of finite strings of symbols from a finite alphabet [24], but they also require considerably more training than semi-formal languages [20], [25]. The main advantage of having defined semantics is that it enables the development of automated tools that can efficiently find problems in requirements [20]. In particular, several formal languages have been tailored to express requirements, such as the Requirements Modeling Language (RML) and the Knowledge Acquisition in Automated Specification (KAOS), which make it possible to precisely state the objectives of software systems. In this sense, formal languages can support the systematic analysis of formalized statements and their associated impact [8], which can be used to reveal missing requirements and inconsistencies, predict behaviors, check for the accuracy of requirements, and also promote the stakeholders' understanding by means of clear semantics [26].

To deal with requirements related to SoS and constituent systems, RE needs to change its focus to the right composition of constituents within the SoS that yields desirable emergent behaviors at runtime [7]. Due to the dynamic nature of SoS, RE is a permanent activity that must be frequently revisited during the SoS life cycle [27], [28]. As a result, SoS have a large and complex set of requirements that have several interdependencies [20], [29]. For this reason, informal notations are insufficient for modeling SoS requirements [20]. Even though formal techniques can be used for modeling requirements related to SoS autonomy, evolution, and emergent behaviors, these techniques usually rely on approaches that lack support for SoS openness as well as unpredictability [8]. In this scenario, the combination of semi-formal and formal techniques can be sought to balance the limitations of each approach, e.g., using formal techniques to model critical parts of the system and semi-formal ones to model non-critical parts [30], [31]. However, a complete landscape about notations and techniques that can be used to model SoS requirements is still missing.

Aiming to address this issue, we present in this article a comprehensive literature review on modeling SoS requirements. In particular, we identify which formal and semi-formal languages and techniques have been used for modeling SoS requirements, and how they have been used. We contextualize the current practice for modeling SoS requirement in regards to

5. In the context of this article, UML is a language and each of its diagrams is a technique. Hence, a technique makes it possible to develop a specific structural/behavioral model of a system, while a language expresses systems in a structure that is defined by a consistent set of rules.

6. SysML is a general-purpose graphical modeling language for representing systems that may include combinations of hardware, software, data, people, facilities, and natural objects [21], [22]

specification style and paradigm, and detail known advantages and limitations for many of them, such as UML, SysML, Prototype Verification System (PVS), and Finite State Machine (FSM). We observe that in spite of several SoS being developed for critical domains, only a few languages and techniques that are used for modeling SoS requirements (e.g., [5], [30], [32], [33]) are currently supported by automated tools, probably impacting the accuracy, correctness, and consistency of SoS requirements.

The remainder of this article is organized as follows. Section 2 introduces the main concepts related to SoS and further explains the challenges for modeling SoS requirements. Section 3 provides an overview of related work. Section 4 presents the research protocol for the identification of studies related to the modeling of SoS requirements. Section 5 presents the main results of our review, and Section 6 concludes this work.

2 An Overview of Systems-of-Systems and Challenges for Modeling Systems-of-Systems Requirements

SoS are formed of heterogeneous and independent constituent systems that work together to perform a mission [1], [3], [34], [35]. SoS present a set of inherent characteristics that were originally identified by Maier [1] and that have been expanded and rewritten since then to fit specific application domains [8], [36]. In addition to the two characteristics discussed in Section 1 (i.e., evolutionary development and emergent behavior), there are three others that refer to the properties of constituent systems, namely: (i) *operational independence*, meaning they have their own functionality even when not cooperating with other constituents; (ii) *managerial independence*, meaning they are independently managed by their owners; and (iii) *geographical distribution*, meaning they interact among themselves exclusively in terms of information exchange, being distributed over different locations. Interactions among constituents can lead to the emergence of new SoS behaviors, functionalities, or missions, which might be triggered or influenced by a stimulus from the system's environment [3]. These emergent behaviors enable SoS to provide functionalities that were not originally designed or that cannot be predicted at design time from knowing only what their constituents are [8], [36].

Constituent systems sometimes have unsynchronized life-cycles. This directly impacts the evolution of an SoS through changes in its fundamental structure, including its constituents and the relationships among them [6], [37]. In such cases, the SoS architecture must be open to such changes and evolve accordingly over time in order to embrace new requirements and situations.

SoS must not restrict the autonomy of constituents, since these constituents can modify their behavior to gain benefits. Moreover, it is necessary to fulfill new requirements and missions of the SoS and also consider the trade-off between constituent missions and SoS missions [37]. In this sense, we still need to better understand the principles that control the constituents' behavior in order to align them with the SoS mission [6]. Appropriate mechanisms must be identified and put in place to support the fulfillment of SoS missions, including both regulating mechanisms, which minimize inappropriate

behavior, and awarding mechanisms, which encourage desirable conduct [6], [7], [37].

The literature often classifies SoS within four different categories that reflect different levels of managerial control and central authority exercised by the SoS over their constituents as well as different levels of collaboration among such constituents [1], [2], [10], [38]. A Virtual SoS has neither central authority to manage its constituents' activity nor a clear purpose; a Collaborative SoS has its constituents working together more or less voluntarily to fulfill agreed central purposes; an Acknowledged SoS has its constituents maintaining their independent ownership, objectives, funding, and development approaches, but has no complete authority over its constituents; and a Directed SoS has central management and an engineering team that builds the SoS aiming to fulfill specific purposes whilst having complete authority over the evolution of its constituents. Unlike the latter two types, the first two have no SoS engineering team guiding or managing activities related to the whole SoS.

Because of their inherent characteristics, especially emergent behavior and evolutionary development, the process for requirements modeling for SoS is extremely challenging and should adequately address real-world SoS problems [39]. Particular attention should be given to understanding stakeholders' demands, interoperability among constituent systems, architecture, and dynamic evolution, as well as to comprehending how emergent behaviors impact requirements stability [9], [40]. Hence, practitioners are sometimes limited in applying traditional approaches (techniques, methods, and tools) to identify the requirements of complex problems in the SoS context. Approaches should be able to handle several problems associated with the requirements of the whole SoS and, at the same time, should be able to deal with the requirements of the various constituent systems evolving independent from each other [40]. In this sense, the use of new concepts and approaches (such as those based on agents, goals, objectives, and actions) could be explored to make SoS requirements modeling more flexible [41].

3 Related Work

To the best of our knowledge, there exists no survey, systematic literature review (SLR), or systematic mapping (SM) on formal and semi-formal languages and techniques for modeling SoS requirements. Nonetheless, this section discusses two SLRs, one concerning modeling requirements languages [42] and the other one concerning specification and methods/techniques for the generation of textual requirements specifications [43]. Afterwards, we will also discuss two works on the comparison of specific formal languages for the description of requirements [44].

Selpúvida et al. [42] analyzed 54 studies to evaluate requirements modeling languages in terms of their maturity, level of expressiveness, origin (i.e., developed in industry, academia, or both), context of use (i.e., industry or academia), and validation. Their work was developed in the context of software product lines. As their main result, the authors point out the lack of a conceptual foundation and tool support for using these languages. Nicolás and Toval [43] evaluated 30 studies, also in the context of software product lines, and report several methods and techniques that can support the generation of

textual requirements specifications from software models (i.e., graphical models). The authors mention tool support as a concern and noticed several difficulties in finding the business requirements and the rationale underlying graphical models. Their work does not classify such methods and techniques with regards to formal or semi-formal issues.

Dutertre and Stavridou [45] present a terminology and a comparative analysis between two formal specification languages, namely Requirements State Machine Language (RSML) and Software Cost Reduction (SCR). Their work compares these languages in the avionics domain with the aim of improving requirements traceability. Thus the focus is not on narrow language characteristics, but rather on performance. Although the authors present a detailed analysis together with information about traceability performance, the work is restricted to these two languages and one application domain, while our SM is more comprehensive and also investigates the combination of languages for requirements modeling, in addition to modeling techniques. Similarly, Sharma and Sing [44] compare the syntax of five formal languages, namely Z, Object Constraint Language (OCL), VDM, Specification and Description Language (SDL), and Larch. The goal of this work at comparison using parameter specification types and mathematical types used to support the choice of an appropriate language for a particular problem.

Finally, we found that all related work focuses on more specific issues, while our work is a wider investigation, analyzing semi-formal and formal ways to model SoS requirements.

4 Planning and Conduction of the Systematic Mapping

Our SM was conducted from April 2017 to August 2017 by software engineering researchers and requirements engineering experts. To conduct this mapping, we followed the systematic process proposed by Petersen et al. [46] and Kitchenham and Charters [47]. In short, this process is composed of three main phases (planning⁷, conduction, and reporting), which will be explained in more detail in the following sections.

4.1 Planning

Aiming to find all relevant primary studies for our research, if possible, we established the following research question (RQ):

RQ: Which formal and semi-formal languages and techniques have been used for modeling SoS requirements?

The search strategy for answering this RQ is based on two main keywords, namely “requirement modeling” and “formal⁸”. To avoid missing out on any relevant study, we also considered the keywords “requirement verification” and “requirement validation” for our search string. Additional terms were also identified together with the opinion of experts in RE and SoS. After a number of refinement iterations and string calibration,

7. The complete research protocol can be found at <https://goo.gl/yntV3m>.

8. The keyword “semi-formal” was not included in the search string, because our string already includes the term “formal”, which allows finding studies related to semi-formal languages/techniques. This was confirmed in our pilot study.

the final search string used in this SM was:

("requirement modeling" OR "model of requirement" OR "requirement model" OR "modeling requirement" OR "modeling of requirement" OR "requirement representation" OR "requirement analysis" OR "analysis of requirement" OR "requirement analyzing" OR "requirement design" OR "requirement verification" OR "verification of requirement" OR "evolution of requirement" OR "requirement evolution" OR "requirement validation" OR "validating requirement" OR "validation of requirement" OR "requirement specification" OR "quality requirement" OR "non functional requirement" OR "non-functional requirement" OR "nonfunctional requirement" OR "non functional property" OR "non-functional property" OR "nonfunctional property" OR "non functional characteristic" OR "non-functional characteristic" OR "nonfunctional characteristic" OR "quality attribute" OR "quality characteristic" OR "quality factor" OR "quality criterion") AND (formal).

It is worth highlighting that the term "system-of-systems" (and its synonyms such as SoS) was not included in our string because this term is not necessarily used by all studies that address SoS. Besides that, other related terms, such as "cyber-physical system", "ecosystem", and "distributed system", could be used to refer to an SoS. Therefore, neither "system-of-systems" nor related terms were included in the string. Hence, the selection of studies that address SoS was conducted by reading each study in its entirety.

To select publication databases for our mapping, we followed specific criteria [47], [48]. The ACM Digital Library, ISI Web of Science, IEEE Xplore, Science Direct, Scopus, and SpringLink databases are the most relevant ones for computer science, and are widely used in software engineering [49], [50], [51]. Our SM was also complemented by manual selection of studies from publication venues that are not indexed by any of these databases.

4.1.1 Inclusion and Exclusion Criteria

We defined specific selection criteria for evaluating primary studies recovered from publication databases. These criteria were applied in the first and second round of selection to identify relevant studies. To include a study in our mapping, we had only one criterion: studies that address formal and/or semi-formal modeling of SoS requirements and similar systems (that have the characteristics of SoS). On the other hand, we had several criteria for excluding a study from our SM. A study was excluded if it is related to the modeling of monolithic systems or if it is an editorial, keynote, opinion, tutorial, panel, extended abstract⁹, or gray literature (e.g., technical report). We also excluded studies where a newer or more complete version exists. For example, study [52] is more complete than study [53], hence only the former is included in our mapping. Besides, studies in languages other than English and those whose full text is not available were also excluded.

4.1.2 Data Extraction and Synthesis Method

The data extracted from each primary study was managed with the support of the Parsif.al¹⁰ and Tableau Public¹¹ tools, and MS Excel. Data extraction was accomplished by the first author

9. For this work, we considered as extended abstracts all studies with up to three pages.

10. <https://parsif.al/>

11. <https://public.tableau.com/s/>

of this article and discussed and reviewed by the other authors. Concordance meetings were also conducted when necessary to discuss data and their relationships. To summarize and present these data, we used qualitative analysis methods.

4.2 Conduction

The primary studies were selected following the protocol briefly described in Section 4.1. 4,751 studies were obtained: 3,993 unique studies were recovered from six publication databases and 758 from manual selection in publication venues. After removing duplicate studies (i.e., 839 of them), 3,912 studies remained for selection. Initially, the title, abstract, and, when necessary, the introduction section of each study were read and the selection criteria were applied. In this way, a total of 96 studies were selected. The full text of each study was then read and the selection criteria were applied again. As a result, a set of 25 primary studies were selected for data extraction. Besides, one study [54] was inserted following the suggestion of an expert, bringing the total to 26 studies. Table 1 presents these studies, their ID, author name (s), publication title that are external link where their study was published, and publication year.

4.3 Threats to Validity

To increase the trustworthiness of our SM and minimize biases that could be introduced by the authors [55], we identified potential threats to validity and discuss below the actions we put in place to mitigate them.

- *Missing important primary studies:* As already considered in Kitchenham and Charters' guidelines [47], it is not possible to guarantee the identification of all relevant primary studies that exist in the literature for a given research topic. To mitigate this threat, we carefully established and validated the research protocol together with experts. We also conducted a pilot study and defined the selection criteria in order to minimize the risk of excluding relevant studies. Besides that, we adopted a specific group of databases considered as the most relevant ones for computer science according to [49], [50], [51]. However, primary studies indexed by other databases could be missing. To mitigate this threat, we also performed a manual search of publication venues and studies recommended by experts consulted for this mapping.
- *Selection of primary studies:* Studies were selected based on how the authors referred to their studies, i.e., modeling, specification, validation, and verification. However, there is a narrow line between these terms/concepts when considering RE. Indeed, in practice, these terms have been used interchangeably and even the context of their application does not make the meaning clear. So, whenever there was doubt about the inclusion of any primary study, it was discussed with experts during the concordance meetings.
- *Number of reviewers and reliability:* Our SM was conducted by three researchers. This number of reviewers may imply some risk of bias. To ensure the reliability of our SM and to obtain an unbiased selection process, our planning phase, including the research question and the selection criteria (i.e., inclusion criterion and exclusion criteria), was carefully set up in advance. The research question and the selection criteria are detailed enough to make it possible to reproduce the steps to obtain the 26 primary studies selected. In

TABLE 1
Primary studies included

ID	Author	Title	Year
S1	Atkinson, W. and Cunningham, J.	Proving properties of a safety-critical systems	1991
S2	Leveson, N.G. and Reese, J.D.	Requirements specification for process-control systems	1994
S3	Heimdahl, M.P.E. and Leveson, N.G.	Completeness and consistency in hierarchical state-based requirements	1996
S4	Crow, J. and Di Vito, B.	Formalizing Space Shuttle software requirements: four case studies	1998
S5	Jong, E. et al.	Refinement in requirements specification and analysis: a case study	2000
S6	Sánchez-Alonso, M. and Murillo, J. M.	Specifying cooperation environment requirements using formal and graphical techniques	2002
S7	Ponsard, C. et al.	Early verification and validation of mission critical systems	2005
S8	Linhares, M. V. et al.	Introducing the modeling and verification process in SysML	2007
S9	Ghazel, M. and El Koursi, E.M.	Automatic level crossings: from informal functional requirements' specifications to the control model design	2007
S10	Jamal, M. and Zafar, N.A.	Requirements analysis of air traffic control system using formal methods	2007
S11	Goldsby, H. J. et al.	Goal-based modeling of dynamically adaptive system requirements	2008
S12	Krishna, A. et al.	Consistency preserving co-evolution of formal specifications and agent-oriented conceptual models	2009
S13	Sun, H. et al.	Automata-based verification of security requirements of composite Web Services	2010
S14	Tang, W. et al.	Scenario-based modeling and verification for CTCS-3 system requirement specification	2010
S15	Tang, W. et al.	Scenario-based modeling and verification of system requirement specification for the European Train Control System	2010
S16	Whittle, J. et al.	RELAX: A language to address uncertainty in self-adaptive systems requirement	2010
S17	Yuan, L. et al.	Modelling and verification of the system requirement specification of train control system using SDL	2011
S18	Cimatti, A. et al.	Formalizing requirements with object models and temporal constraints	2011
S19	Zhang, L.	Aspect-oriented formal techniques of cyber physical systems	2012
S20	Deb, N. and Chaki, N.	Verification of i* models for existential compliance rules in remote healthcare systems	2014
S21	Zhang, L.	Modeling large scale complex cyber physical control systems based on system of systems engineering approach	2014
S22	Zou, L. et al.	Verifying Chinese train control system under a combined scenario by theorem proving	2014
S23	Chen, Z. et al.	Exploring a timed-automata fuzzy cognitive maps based approach for modeling systems of systems	2015
S24	Piccolo, A. et al.	Use of formal languages to represent the ERTMS/ETCS system requirements specifications	2015
S25	Han, L. et al.	Safety requirements specification and verification for railway interlocking systems	2016
S26	Wang, Q.-L. et al.	A quality requirements model and verification approach for system of systems based on description logic	2017

addition, SoS and RE experts supported the entire selection process through concordance meetings.

- *Non-available studies and data extraction:* 88 primary studies obtained in the databases (i.e., 2.2% or 88/3,993) were not available. Despite efforts to contact the authors by email or via social network such as ResearchGate¹², we did not gain access to them. Moreover, some information described in the included studies was not clear and had to be interpreted. To ensure the validity of our SM, discussions with experts were carried out whenever there were doubts.

5 Results

The following sections report the findings of our SM. First, Section 5.1 presents an overview of primary studies that were used for answering our RQ in Section 5.2. Our main findings are then discussed in Section 5.3.

5.1 General Results

Table 2 shows an overview of the 26 primary studies included in our SM. Taking into the account challenges for modeling SoS requirements (cf. Section 2), we can already observe several initiatives that use formal and/or semi-formal languages

for addressing these issues. In fact, 80.8% of the included studies model SoS requirements by means of formal languages or techniques. This choice can be justified by the critical application domains in which these studies were performed and also by regulations such as DO278 [56] and ISO/DIS 26262 [57], which require specific procedures for the certification of these systems.

Our SM revealed the use of semi-formal languages and techniques in eleven studies, of which six apply a mix of formal and semi-formal languages and techniques (i.e., S6, S7, S8, S9, S12, and S21). Formal languages and techniques are specifically used in these studies for modeling critical parts of the system (e.g., brakes or rudder control systems of an airplane) whilst non-critical parts have been modeled using semi-formal languages [56], [58]. The remaining five studies (i.e., 5 of 11) only used semi-formal languages (i.e., S11 and S20) or semi-formal techniques (i.e., S14, S15, and S18) for modeling SoS requirements. For instance, S14 [59] and S15 [60] apply UML for modeling scenario-based requirements of a train control system. In particular, the authors use Sequence Diagrams (UML-SD) for the representation of interactions and behaviors of this system and its components exhibited in operations.

Overall, the 26 studies included in this SM cover 12

12. <https://www.researchgate.net/>

TABLE 2
Characterization of the included primary studies

		Primary Studies																										Total
		S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	
Modeling	Formal	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	21
	Semi-formal						✓	✓	✓	✓		✓	✓		✓	✓			✓		✓	✓						11
Application Domain	Aerospace				✓																							1
	Airspace										✓																	1
	Automotive																					✓						1
	Avionics		✓	✓																								2
	Defense																										✓	1
	Factory								✓																			1
	Healthcare																						✓					1
	Natural Disaster											✓	✓	✓														3
	Naval					✓																						1
	Railway	✓						✓		✓					✓	✓		✓	✓	✓			✓		✓	✓		11
	Smart Home															✓												1
	Water Treatment																							✓				1
	No defined					✓																						1

different critical application domains, as depicted in Table 2. The most thoroughly explored domain is railway systems, which is addressed by 42.31% (11 studies) of the included studies, followed by natural disaster (3 studies) and avionics (2 studies). Only one included study, S6, does not focus on any particular application domain. In S6, the authors investigated formal and graphic techniques for complex systems in general, but evaluated their work in the context of the fire control system for a museum. All domains identified in this SM are safety-critical, i.e., system failures can harm people and/or the environment, or result in significant financial loss. As SoS have become increasingly predominant in safety-critical domains, as observed in this SM, more rigor is certainly needed to properly deal with requirements, which is reflected in the greater interest in formal approaches.

5.2 RQ: Formal and Semi-formal Languages and Techniques for SoS Requirements Modeling

Different formal and semi-formal languages and techniques have been adopted to model SoS requirements. Some identified languages are variants of others, e.g., Timed-CSP is a variant of the parallel language Communication Sequential Process (CSP). In particular, we identified 19 languages in the 26 included studies, of which 15 are formal and five are semi-formal. We also identified 13 techniques, of which six support formal modeling and seven support semi-formal modeling. These languages and techniques are represented in Figure 1. In this figure, number of studies included per year is reflected in the size of the circles that represent years. For instance, the year 2010 is represented by a larger circle than the year 1991 since it contains four studies, whereas the latter contains only one. Formal languages and techniques are represented in the top (blue) region of this figure, while semi-formal languages and techniques are shown in the bottom (yellow) region. We also distinguished modeling languages, which are shown in black font, from modeling techniques, which are shown in red font. For instance, the year 2010 contains two studies (i.e., S14 and S15) that use semi-formal languages (i.e., UML-SD), one study (i.e., S13) that uses a formal language, and two studies (i.e., S13 and S16) that use formal techniques. In particular, S13

combines three formal languages, namely eXtensible Markup Language (XML), Web Service Definition Language (WSDL), and Web Service Business Process Execution Language (WS-BPEL), in addition to using the FSM technique, and finally, S16 adopted only Fuzzy Logic.

Formal and semi-formal languages and techniques can be classified and compared in terms of specification styles, paradigm, and executable syntax. This comparison was done according to the specification styles described in [61], [62], [63], [64], [65]. The most important characteristic of the specification language is based on its mathematical foundation [61]. However, different terminologies are used in each study for the same style. For example, the technical report of NASA [62], [63] classifies languages and techniques as either model-oriented or property-oriented. The former can be considered as a constructive (or prescriptive) style, whereas the latter can be considered as a declarative (or descriptive) style, which is also referred to as axiom-based or rule-based by other studies [64], [65]. To clarify the classification of specification styles used in this SM, we created the conceptual model presented in Figure 2. We refer to this conceptual model in the column “Specification Style” in Table 3.

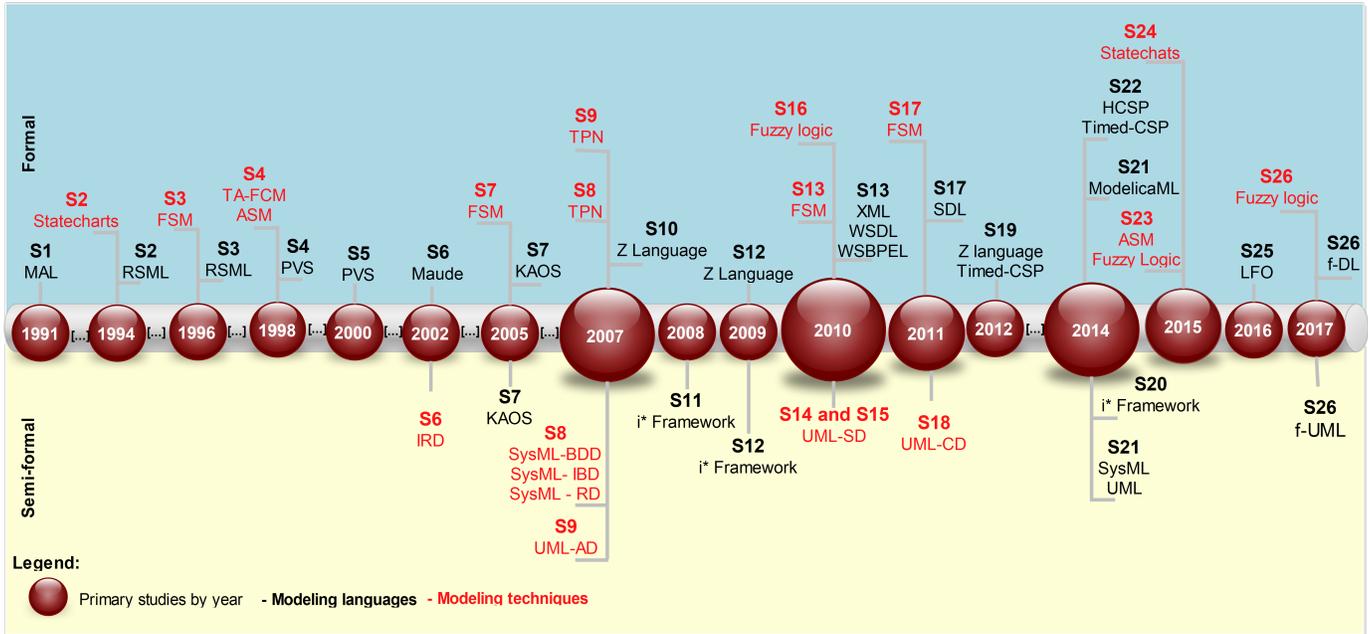
A model-based specification, also known as constructive and prescriptive style, describes the desired behavior for an intended system usually by means of models [65]. While this specification style is often able to create detailed models of the system, an excess of information can also lead to bias in design and implementation [63]. Nonetheless, model-based specifications are often easier for non-technical users to understand than property-oriented specifications. The latter can be used to describe desired properties of a system at a higher abstraction level, which generally leads to a specification with fewer details [62], [65]. On the other hand, it is easier to introduce inconsistencies in this style, which also requires advanced knowledge to be read and understood by users [62], [63], [65]. A detailed discussion on the trade-offs between the two specification styles can be found in [62], [63].

Other characteristics analyzed in Table 3 include:

- *Paradigm*: classifies a language or technique according to modeling features;

TABLE 3
Classification of formal and semi-formal languages and techniques

	Description Languages/Techniques	Specification Style	Paradigm	Graphical Model	Formalism	Language ● / Technique ⊙	Executable	Target	Documentation
Model-Based Specification	Object-Oriented	Prescriptive/Logic-basic	Object-oriented	√	S	⊙		Domain-Specific modeling to systems engineering	√
					S	⊙		General	√
					S	⊙		General	√
					S	⊙		General	√
	S	⊙		Domain-Specific modeling to systems engineering	√				
	S	⊙		Domain-Specific modeling to systems engineering	√				
	S	●	√	Large-Scale Complex Systems/ Systems-of-Systems	√				
	S	●		Domain-Specific modeling to systems engineering	√				
	S	●		General	√				
	S	●		General	√				
State-Based	Specification and Description Language - SDL	Prescriptive	State-based	√	F	●	√	Large Real-Time Systems	√
					F	●	√	Safety-Critical Systems	√
					F	●		General	√
					F	●		Concurrent and Real-Time Systems	√
	F	●		Hybrid Systems	√				
	F	●	Event-based				√	Cyber Physical Systems	√
	F	●			√		√	Real-Time Systems	√
	F/S	●	Agent-based		√		√	General	√
	S	●			√		√	General	√
	Hybrid	ModelicaML	Prescriptive	Object-oriented/Time-discrete or Event-based	√	F	●	√	Systems-of-Systems/Large Scale Complex Cyber Physical Control Systems
F						⊙	√	Systems-of-Systems/Large and Complex Systems	√
Timed-Automata-Based Fuzzy Cognitive Maps - TA-FCM		Prescriptive	Concepts-based/State-based	√	F	⊙	√	Systems-of-Systems/Large and Complex Systems	√
					F	⊙	√	Safety-Critical Systems/Real-Time Systems	√
Abstract State Machine - ASM		Declarative	Rules-based			⊙		SoS/Self-Adaptive Systems	√
						●	Interpreted	General	√
Finite State Machine - FSM		Prescriptive	Object-oriented/State-based	√		⊙	√	General	√
						●	√	General	√
Statecharts		Declarative	XML-based	√		⊙	√	Real-Time Embedded Systems	√
						●	√	General	√
Interleaved Requirements Diagram - IRD	Declarative	XML-based	√		⊙	√	General	√	
					●	√	General	√	
Fuzzy Logic	Declarative	XML-based	√		⊙	√	General	√	
					●	√	General	√	
eXtensible Markup Language - XML	Declarative	XML-based	√		⊙	√	General	√	
					●	√	General	√	
Formal Language Maude	Declarative	XML-based	√		⊙	√	General	√	
					●	√	General	√	
Logic First Order - LFO	Declarative	XML-based	√		⊙	√	General	√	
					●	√	General	√	
Modal Action Logic - MAL	Declarative	XML-based	√		⊙	√	General	√	
					●	√	General	√	
Web service definition language - WSDL	Declarative	XML-based	√		⊙	√	General	√	
					●	√	General	√	
Web Service Business Process Execution Language - WSBPEL	Declarative	XML-based	√		⊙	√	General	√	
					●	√	General	√	
Property-Based Specification	Fuzzy Description Logic - FDL	Declarative	Knowledge-based		F	●	√	Large-Scale Complex Systems/ Systems-of-Systems	√
					F	●	√	Large-Scale Complex Systems/ Systems-of-Systems	√



Modeling Languages and Techniques Legend:
 Abstract State Machine - ASM; eXtensible Markup Language - XML; Finite State Machine - FSM; Formal Language Maude - Maude; Fuzzy Description Logic - f-DL; Fuzzy Unified Modeling Language - f-UML; Hybrid Communicating Sequential Processes - HCSP; i* Framework; Interlelement Requirements Diagram - IRD; KAOS Methodology - KAOS; Logic First Order - LFO; Modal Action Logic - MAL; Prototype Verification System - PVS; Requirements State Machine Language - RSM; Specification and Description Language - SDL; SysML Block Definition Diagram - SysML-BDD; SysML Internal Block Diagram - SysML-IBD; SysML Requirements Diagram - SysML-RD; Timed Communicating Sequential Processes - Timed-CSP; UML Activity Diagram - UML-AD; UML Class Diagram - UML-CD; UML Sequence Diagram - UML-SD; Web Service Business Process Execution Language - WSBPEL; Web Service Definition Language - WSDL.

Fig. 1. Overview of the formal and semi-formal techniques and languages per year

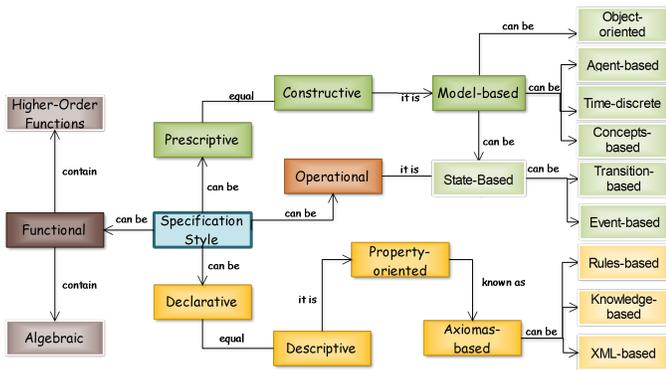


Fig. 2. Conceptual model of specification style terminologies

- *Graphical Model*: indicates if a language or technique supports a graphical notation;
- *Formalism*: indicates whether a language or technique is formal (F) or semi-formal (S);
- *Language/Technique*: indicates whether it is a language or technique;
- *Executable*: indicates whether is possible to execute the models created using the language/technique for the purpose of requirements V&V, which could minimize time and costs [66];
- *Target*: indicates which sorts of systems are described using the language or technique (e.g., large real-time system or systems-of-systems); and
- *Documentation*: indicates the existence of external information for the language or technique. For example, f-UML is only described in the included studies whereas the Z language is also described by the ISO/IEC13568:2002

standard [67].

5.2.1 Model-based Specification

Our SM identified five model-based specification paradigms: object-oriented, state-based, event-based, agent-based (also known as goal-based), and hybrid¹³. The hybrid paradigm encompasses one language, i.e., ModelicaML, and four techniques, i.e., statecharts, FSM, abstract state machines, and Timed-Automata-based Fuzzy Cognitive Maps (TA-FCM). The object-oriented paradigm has a great impact on systems development and, consequently, on requirements modeling. Our SM identified the use of UML and its variants such as SysML. For instance, S21 combines UML and SysML to ModelicaML, which is deriveds from UML and SysML, to model the requirements of a cyber physical SoS. In particular, S8 combines the Requirements Diagram, the Block Definition Diagram, and the Internal Block Diagram of SysML with Time Petri Nets for modeling and model checking the behavior requirements of a factory plant. UML techniques, specifically Activity Diagram, Class Diagram, and Sequence Diagram, are used for representing requirements in S8, S8, S14, S15, and S18. Even though our SM identified UML and SysML as recurrent languages for the development of object-oriented systems in industry, these languages and their variants such as executable UML (xUML¹⁴) still lack formal execution semantics that can support formal modeling. For this reason, we have also noticed their combination with formal approaches [68].

Most included studies combine two or more languages and techniques for modeling SoS requirements. Particularly, S2, S3, S4, and S17 combine formal languages and formal

13. We classify as a hybrid paradigm those languages and techniques that can be classified in more than one paradigm.

14. <https://xtuml.org/>

techniques; S7 and S12 combine formal languages and semi-formal techniques; S8 and S14 combine formal and semi-formal techniques; and S19, S21 and S22 combine formal and semi-formal languages. For instance, S17 employs one formal language, i.e., the Specification and Description Language (SDL), and one formal technique, i.e., Finite State Machines (FSM), in a method for the modeling and verification of the systems requirements of a train control system. This leads us to infer that SoS requirements modeling is not completed using a single formal language or technique but instead requires a set of languages and techniques to properly address the modeling challenges posed by each domain. In this sense, more studies are needed to investigate SoS requirements modeling in other domains, considering the different challenges that might be faced in each of them, and to evaluate which formal and/or semi-formal languages and techniques provide better support for SoS inherent characteristics.

Our SM identified two languages that support agent-based paradigms in industrial environments [69], [70], namely the i^* framework and the KAOS methodology. Agents can be defined as active components representing people, devices, legacy software, or software-to-be, which are responsible for fulfilling specific requirements and expectations [71]. The i^* framework is a conceptual modeling language adopted by S11, S12, and S20. This language is used for modeling the environment of a system-to-be, supporting critical modeling decisions, such as identification of the main system's goals, representation of multiple stakeholders and their interdependence, and possibilities for exploring the use of these relationships [52], [72]. These characteristics can be interesting for SoS requirements modeling since SoS are formed by interacting independent constituents that individually address a particular set of stakeholders and goals [1], [6], [73]. Aside from agents, the KAOS methodology adopted by S7 represents goals that express desired system properties stated by stakeholders that are met by agents [71]. In general, the language supports four techniques: (i) goal model: forms a set of interrelated goal diagrams that have been put together to address a particular problem; (ii) object model: describes objects (e.g., agents, entities, and relationships); (iii) responsibility model: it describes which requirements and expectations an agent is assigned to; and (iv) operation model: describes all behaviors that agents need to perform to fulfill their requirements [30], [71]. KAOS assists in the establishment of both formal and semi-formal models. Semi-formal models are based on text and include graphical representations, whereas formal models are built on top of semi-formal models, either partially or entirely [71].

The state-based paradigm can be considered in the behavioral paradigm, since it is used to represent systems behavior by transitioning among different states. In critical systems, models depicting behavior diagrams such as FSM and sequence diagrams, have been employed not only in the modeling of requirements for safety-critical systems but also in their verification by means of simulations. Simulations are traditionally used for evaluating different execution scenarios at design time and, more recently, have also been used for simulating SoS [74], [75]. However, SoS simulation is complicated due to a combination of factors, such as performance issues, conflicting goals, standards, and emergent behaviors (which might be known or unknown at design time) [74], [76]. In spite of this

difficulty, simulation also brings important benefits such as early identification of errors and problems that can be corrected before the actual realization of the SoS [75]. Languages and techniques supporting the state-based paradigm were used by 52% of the included studies (i.e., S2, S3, S4, S5, S7, S10, S12, S13, S17, S19, S23, S24, and S26). An example is S8, which used PVS, a language with tool and theorem proof integrated with decision procedures for different theories including real and integer arithmetics, in conjunction with an abstract state machine and TA-FCM. This study investigates the formalization of subsystem modeling of NASA's Space Shuttle using PVS to explore and document the feasibility and utility of formalizing critical Shuttle software requirements representing a spectrum of maturity levels. PVS is still being used at NASA Langley Research Center (LaRC) for modeling requirements of aerospace applications, such as the pilot flying specification [58] and the aerospace verification tool [77].

Although we have classified PVS as a model-based style (i.e., state-based paradigm), it can also be considered a property-based style (i.e., axiom-based paradigm), since a set of properties are described to ensure consistency of the specification [62], [63]. In addition, it also considers higher-order logic or higher-order function, which is classified as a functional style [64]. The same author classifies Time Petri Net as an operational style, while others classify it as a model-based style [62], [63], [65].

5.2.2 Property-based Specification

Our SM also identified three property-based paradigms: (i) the rule-based paradigm, which uses formal and semi-formal languages and techniques; (ii) the XML-based paradigm, which uses formal languages; and (iii) the knowledge-based paradigm, which uses formal languages. Even though there are fewer property-based paradigms, they have been used in SoS requirements modeling [58], [78], [79], [80]. All identified property-based languages, namely XML, WSDL, WS-BPEL, FOL, Maude, Modal Action Logic (MAL), and fuzzy Description Logic (f-DL), and property-based techniques, namely Interelement Requirements Diagram (IRD) and Fuzzy Logic, were sometimes applied separately by each identified study. For instance, S26 adopted the Description Logic ontology for the description of the quality requirements, which is a logical reconstruction of a frame-based knowledge representation language. To specify functional and non-functional SoS requirements that are considered fuzzy and vague requirements at the mission level an extension of f-DL called f-SHIN was applied to describe the necessary quality of the requirements, as it has a strong ability of representation and decidability. f-DL is introduced to formalize the UML model and to provide an algorithm for converting a fuzzy UML (f-UML) model into the f-DL ontology to automate verification.

With regard to SoS requirements verification, S25 focus on automation and verification of safety requirements based on pattern-based specification. These requirements were specified using First Logical Order (FOL), which belongs to the rule-based paradigms, to verify safety requirements in the railway domain automatically through model checking. Due to the characteristics of SoS, the creation of an environment for verifying requirements is challenging. Each constituent has its own verification responsibility; therefore changes in constituents may result in verification events that might affect

the entire SoS [74]. Moreover, the managerial independence of the constituents often does not allow synchronization of multiple life cycles. Hence, verification of SoS requirements and of the SoS itself sometimes occurs without the presence of all the constituents participating in an SoS (i.e., without all capabilities) [41], [81]. Besides that, due to the evolutionary development of SoS, requirements modeling and verification should be performed continually, including evaluation of the system's capability regarding its missions [74], [82].

5.3 Discussion

Our investigations suggest that formal and semi-formal modeling of SoS requirements is an essential activity for developing these systems that requires more rigor in the requirements specification process. Hence, applying more formal approaches in SoS modeling and V&V can increase the consistency, correctness, and completeness of their requirements. Similarly, industrial standards have also reinforced this necessity, mainly by means of certification of these systems. Example as, the DO278 [56] used in the certification of avionics, IEC/TR 80002 [83] used in medical device development, and EN 50126 [84], ISO/DIS 26262 [57] applied in road vehicles, and EN 50128 [85] used in railway applications.

Our SM draws on 26 studies, selected out of 4,751 using a systematic process composed of several stages. All 26 studies address formal and/or semi-formal languages and/or techniques. An important feature of our SM is that, although we focused on the SoS domain, we did not narrow our search by including SoS in our search string, which allowed us to search more deeply for information regarding the state of the art in formal and semi-formal modeling of SoS requirements. We found 15 studies modeling part or all SoS requirements using only formal languages or techniques, five adopting only semi-formal ones, and another six combining formal and semi-formal ones.

SoS have an inherent complexity in their development with new requirements emerging at runtime and a multiplicity of constituents, which are sometimes unknown at design time or can change at runtime. Modeling the requirements of these SoS and their constituents certainly requires a combination of formal and/or semi-formal languages and techniques. The adoption of formal modeling in all systems requirements may increase the development cost and the time spent on specification, but could minimize errors introduced in other lifecycles phases. Besides that, formal languages and techniques offer the precision necessary for modeling the requirements of critical, complex systems; however, they require mathematical expertise from their users. A common strategy is to encapsulate the formal information in specification tools, which minimizes the need by for mathematical expertise, maintains the precision of the requirements specifications, and could encourage adoption the industry. On the order hand, modeling using only semi-formal languages and techniques may reduce the time needed to produced a specification, but might increase the number of errors introduced in the development process.

We identified 32 formal and semi-formal techniques and languages, of which 19 are executable and one language (i.e., XML) is interpreted. In other words, requirements specified in these 20 techniques/languages can be evaluated automatically with tool support. Tools can increase the precision of analyses

and the accuracy of the completeness, consistency, and correctness of SoS requirements. Moreover, 25 languages/techniques from the 38 one found are well documented (cf. Table 3, which might facilitate their adoption. However, the research field is relatively new to SoS and some studies from other domains, like embedded systems, cyber-physical systems, and self-adaptive systems, were found that propose ideas that may be adapted to SoS.

A deeper analysis of formal and semi-formal V&V of SoS requirements still needs to be performed. We also observed that most of these approaches have been applied to systems with characteristics compatible with SoS, but not exactly SoS, considering their inherent characteristics. Our investigation only revealed three (S21, S23, and S426) where these characteristics (i.e., managerial and/or operational independence, evolutionary development, emergent behavior, and geographical distribution) are addressed explicitly. Hence, we believe that more attention still needs to be paid to SoS RE, specifically regarding modeling and verification, which are activities that ensure high-quality requirements. Finally, tools that consider not only requirements at the SoS level, but also the elicitation, modeling, and verification of the constituents' requirements need to be developed in a more integrated way.

6 Conclusions

Formal and semi-formal modeling of SoS requirements can certainly contribute to improving the quality of these systems, mainly when applied together with requirements verification. Even considering the advantages of formal modeling for critical systems development, the use of formal modeling in the industrial sector is still hindered by two specific issues: (i) the cost of formal application is high; and (ii) specialized professionals are required to understand the semantics of specification languages. In this scenario, the main contribution of this work is a landscape of languages and techniques for formally and semi-formally modeling SoS requirements, including initiatives adopted in similar systems that could be useful for SoS. For this purpose, we applied a systematic approach to the identification and analysis of the studies.

As future work (besides of the needs already mentioned) we intend to perform: (i) a more specific investigation in this research area, for instance, the way that formal and semi-formal V&V of SoS requirements have been addressed and integrated with modeling; and (ii) consolidation of the results of this SM, aimed at providing a more detailed analysis of all the evidence presented in this work. Finally, this work is intended to direct the attention of researchers and practitioners to the importance of adequately treating requirements modeling, particularly when such systems are as complex, critical, and software-intensive as SoS.

Acknowledgments

This work is supported by the São Paulo Research Foundation (FAPESP) under grant no.2015/06195-3, no.2017/15354-3, and no.2017/06195-9. We also thank the Software Architecture Research Team (START) of ICMC/USP, Brazil, for its advice and guidance in improving this work, the anonymous reviewers for their thorough reviews, and Sonnhild Namingha from the Fraunhofer (IESE), Kaiserslautern, Germany, for precious advice and language review.

References

- [1] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998.
- [2] J. Dahmann and K. Baldwin, "Understanding the current state of us defense systems of systems and the implications for systems engineering," in *2nd Annual IEEE International Systems Conference (SysCon 2008)*. Montreal, Canada: IEEE, 2008, pp. 1–7.
- [3] M. W. Maier, "Modeling and simulation support for system of systems engineering applications," L. B. Rainey and A. Tolk, Eds. Hoboken, NJ: John Wiley & Sons. Inc., 2014, ch. The Role of Modeling and Simulation in System of Systems Development, pp. 11–41.
- [4] P. Dersin and A. Transport, "Systems of systems," <http://rs.ieee.org/component/content/article/9/77-system-of-systems.html>, IEEE-Reliability Society, 2014, [Online, Accessed: December 18, 2017]. [Online]. Available: <http://rs.ieee.org/component/content/article/9/77-system-of-systems.html>
- [5] Q.-L. Wang, Z.-X. Wang, T.-T. Zhang, and W.-X. Zhu, "A quality requirements model and verification approach for system of systems based on description logic," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 3, pp. 346–361, 2017.
- [6] S. Y. Han and D. DeLaurentis, "Development interdependency modeling for system-of-systems(sos) using bayesian networks: Sos management strategy planning," in *Conference on Systems Engineering Research (CSER 2013)*. Atlanta, GA: Elsevier, 2013, pp. 698–707.
- [7] C. Ncube, S. L. Lim, and H. Dogan, "Identifying top challenges for international research on requirements engineering for systems of systems engineering," in *21st IEEE International Requirements Engineering Conference (RE 2013)*. Rio de Janeiro, Brazil: PUC-RIO, 2013, pp. 342–344.
- [8] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Pleska, "Systems of systems engineering: basic concepts, model-based techniques, and research directions," *ACM Computing Survey*, vol. 48, no. 2, pp. 18:1–18:41, 2015.
- [9] C. B. Keating, J. J. Padilha, and K. Adams, "System of systems engineering requirements: Challenges and guidelines," *Engineering Management Journal*, vol. 20, no. 4, pp. 24–31, 2015.
- [10] K. Baldwin, "System engineering guide for systems-of-systems engineering," Department of Defense of United States, Tech. Rep., 2008.
- [11] D. A. DeLaurentis, W. A. Crossley, and M. Mane, "Taxonomy to guide systems-of-systems decision-making in air transportation problems," *Journal of Aircraft*, vol. 48, no. 03, pp. 760–770, 2011.
- [12] L. Han, J. Liu, T. Zhou, J. Sun, and X. Chen, "Safety requirements specification and verification for railway interlocking systems," in *40th IEEE Annual Computer Software and Applications Conference (COMPSAC 2016)*. Atlanta, USA: IEEE, 2016, pp. 335–340.
- [13] R. V. D. Watt and L. Erasmus, "Healthcare facility commissioning - the transition of clinical services," in *26th Annual INCOSE International Symposium (IS 2016)*. Scotland, UK: INCOSE, 2016, pp. 1–13.
- [14] ISO/IEC/IEEE 24765, "Systems and software engineering vocabulary," ISO/IEC/IEEE, Tech. Rep., 2017.
- [15] A. A. Alshazly, A. M. Elfatraty, and M. S. Abougabal, "Detecting defects in software requirements specification," *Alexandria Engineering Journal*, vol. 53, no. 3, pp. 513–527, 2014.
- [16] A. Randell, E. Spellman, W. Ulrich, and J. Wallk, "Leveraging business architecture to improve business requirements analysis," Business Architecture Guild, Tech. Rep., 2014.
- [17] B. W. Boehm, *Software engineering economics*. Prentice Hall PTR, 1981.
- [18] S. Maalem and N. Zarour, "Challenge of validation in requirements engineering," *Journal of Innovation in Digital Ecosystems*, vol. 3, no. 1, pp. 15–21, 2016.
- [19] V. Chapurlat, B. Kamsu-Foguem, and F. Prunet, "A formal verification framework and associated tools for enterprise modeling: Application to ueml," *Computers in Industry*, vol. 57, no. 2, pp. 153–166, 2006.
- [20] Q. Wang, Z. Wang, T. Zhang, and W. Zhu, "A quality requirements model and verification approach for system of systems based on description logic," *Frontiers of Information Technology & Electronic Engineering*, pp. 1–17, 2016.
- [21] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: the systems modeling language*. Elsevier, 2008.
- [22] Object Management Group, "Systems modeling language," Available at <http://www.omg-sysml.org/>, 2016, [Online, Accessed: December 06, 2016].
- [23] R. Ahmed and S. Robinson, "Simulation in business and industry: How simulation context can affect simulation practice?" in *Spring Simulation Multiconference (SpringSim 2007)*. Norfolk, Virginia: Society for Computer Simulation International, 2007, pp. 152–159.
- [24] G. Rozenberg and A. Salomaa, *Handbook of Formal Languages*. Springer, 2004, no. 1-3.
- [25] J. V. Gutttag and J. J. Horning, *Larch: Languages and Tools for Formal Specification*. Springer-Verlag New York, Inc., 1993.
- [26] M. Jarke, P. Loucopoulos, K. Lyttinen, J. Mylopoulos, and W. Robinson, "The brave new world of design requirements," *Information Systems*, vol. 36, no. 7, pp. 992–1008, 2011.
- [27] B. W. Boehm, "Verifying and validating software requirements and design specifications," *IEEE Software*, vol. 1, no. 1, pp. 75–88, 1984.
- [28] H. A. Bila, M. Ilyas, Q. Tariq, and M. Hummayun, "Requirements validation techniques: An empirical study," *International Journal of Computer Applications*, vol. 148, no. 14, pp. 5–10, 2016.
- [29] S. Easterbrook, R. Lutz, R. Covington, J. Kelly, Y. Amo, and H. D., "Experiences using formal methods for requirements modeling," NASA/WVU Software Research Lab, Tech. Rep., 1998.
- [30] C. Ponsard, P. Massonet, A. Rifaut, J. F. Molderez, A. van Lamsweerde, and H. Tran Van, "Early verification and validation of mission critical systems," *Electronic Notes in Theoretical Computer Science (ENTCS)*, vol. 133, no. 31, pp. 237–254, 2005.
- [31] L. Zhang, "Modeling large scale complex cyber physical control systems based on system of systems engineering approach," in *20th International Conference on Automation and Computing (ICAC 2014)*. Cranfield, United Kingdom: IEEE, 2014, pp. 55–60.
- [32] W. Atkinson and J. Cunningham, "Proving properties of a safety-critical system," *Software Engineering Journal*, vol. 6, no. 2, pp. 41–50, 1991.
- [33] L. Yuan, T. Tang, and K. Li, "Modelling and verification of the system requirement specification of train control system using sdl," in *10th International Symposium on Autonomous Decentralized Systems (ISADS 2011)*. Washington, USA: IEEE Computer Society, 2011, pp. 81–85.
- [34] D. A. DeLaurentis, "A taxonomy-based perspective for systems of systems design methods," in *IEEE International Conference on System, Man and Cybernetics (SMC 2005)*. Waikoloa, USA: IEEE, 2005, pp. 86–91.
- [35] J. A. Lane, "What is a system-of-system and why should i care?" University of Southern California, Tech. Rep., 2013.
- [36] D. Firesmith, "Profiling systems using the defining characteristics of systems of systems (SoS)," Carnegie Mellon University, Tech. Rep., 2010.
- [37] J. Axelsson, "Systems-of-systems for border-crossing innovation in the digitized society: a strategic research and innovation agenda for sweden," INCOSE Sweden, Tech. Rep., 2015.
- [38] A. Madni and M. Sievers, "System of systems integration: key considerations and challenges," *Journal Systems Engineering*, vol. 17, no. 3, pp. 330–347, 2014.
- [39] R. G. Walker, "A method to define requirements for system-of-systems," PhD Thesis, Faculty of Old Dominion University, USA, 2014.
- [40] C. Ncube, "On the engineering of systems of systems: Key challenges for the requirements engineering community," in *Workshop on Requirements Engineering for Systems, Services and Systems-of-Systems (RESS 2011)*. Trento, Italy: IEEE, 2011, pp. 70–73.
- [41] G. Lewis, E. Morris, P. Place, S. Simanta, and D. Smith, "Requirements engineering for systems of systems," in *3rd Annual IEEE International Systems Conference (SysCon 2009)*. Vancouver, Canada: IEEE, 2009, pp. 247–252.
- [42] S. Sepúlveda, A. Cravero, and C. Cachero, "Requirements modeling languages for software product lines: A systematic literature review," *Information and Software Technology*, vol. 69, no. C, pp. 16–36, 2016.
- [43] J. Nicolás and A. Toval, "On the generation of requirements specifications from software engineering models: A systematic

- literature review,” *Information and Software Technology*, vol. 51, no. 9, pp. 1291–1307, 2009.
- [44] A. K. Sharma and M. Singh, “Comparison of the formal specification languages based upon various parameters,” *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 11, no. 5, pp. 37–39, 2013.
- [45] B. Dutertre and V. Stavridou, “Avionics systems requirements: A comparison of rsm1 and scr,” in *Irish Signals and Systems Conference (ISSC 1998)*. Dublin, Ireland: University of Limerick, 1998, pp. 1–10.
- [46] K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering,” *Information and Software Technology*, vol. 64, pp. 1–18, 2015.
- [47] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” Keele University and Durham University Joint Report, Tech. Rep., 2007.
- [48] O. Dieste, A. Grimán, and N. Juristo, “Developing search strategies for detecting relevant experiments,” *Empirical Software Engineering*, vol. 14, no. 5, pp. 513–539, 2009.
- [49] K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering,” *Information and Software Technology*, vol. 64, no. C, pp. 1–18, 2015.
- [50] B. Kitchenham, D. I. K. Sjøberg, O. P. Brereton, D. Budgen, T. Dybå, M. Höst, D. Pfah, and P. Runeson, “Can we evaluate the quality of software engineering experiments?” in *4th International Symposium on Empirical Software Engineering and Measurement (ESEM 2010)*. Bolzano, Italy: ACM, 2010, pp. 1–8.
- [51] H. Munir, M. Moayyed, and K. Petersen, “Considering rigor and relevance when evaluating test driven development: a systematic review,” *Information and Software Technology*, vol. 56, no. 4, pp. 375–394, 2014.
- [52] A. Krishna, S. A. Vilkomir, and A. K. Ghose, “Consistency preserving co-evolution of formal specifications and agent-oriented conceptual models,” *Information and Software Technology*, vol. 51, no. 2, pp. 478–496, 2009.
- [53] —, “A case study of combining i* framework and the z notation,” in *6th International Conference on Enterprise Information Systems (ICEIS 2004)*. Porto, Portugal: CiteSeer, 2004, pp. 192–200.
- [54] L. Zhang, “Aspect-oriented formal techniques of cyber physical systems,” *Journal of Software*, vol. 7, no. 4, pp. 823–834, 2012.
- [55] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. j. Regnell, and A. Wessln, *Experimentation in software engineering*. Springer Publishing Company, Incorporated, 2012.
- [56] EUROCAE DO278, “Guidelines for communication, navigation, surveillance, and air traffic management (cns/atm) systems software integrity assurance,” The European Organisation for Civil Aviation Equipment, France, Tech. Rep., 2002.
- [57] ISO/DIS 26262, “Road vehicles — functional safety,” International Organization for Standardization, Tech. Rep., 2009.
- [58] D. Cofer, S. P. Miller, and I. C. R. Collins, R., “Formal methods case studies for do-333,” NASA Center for Aerospace Information, Hanover, Tech. Rep., 2014.
- [59] W. Tang, B. Ning, T. Xu, and L. Zhao, “Scenario-based modeling and verification for ctc3 system requirement specification,” in *2nd International Conference on Computer Engineering and Technology (ICCET 2010)*, vol. 1. Chengdu, China: IEEE, 2010, pp. V1–400–V1–403.
- [60] —, “Scenario-based modeling and verification of system requirement specification for the european train control system,” *WIT Transactions on the Built Environment*, vol. 114, pp. 759–770, 2010.
- [61] V. B. Misic and D. M. Velasevic, “Formal specification in software development: A overview,” *Yugoslav Journal of Operations Research*, vol. 7, no. 1, pp. 79–96, 1997.
- [62] NASA, “Formal methods specification and analysis guidebook for the verification of software and computer systems - volume ii: A practitioner’s companion,” NASA OClO, USA, Tech. Rep., 1997.
- [63] M. K. Srivas and S. P. Miller, “Formal verification of an avionics microprocessor,” NASA, USA, Tech. Rep., 1995.
- [64] A. v. Lamsweerde, “Formal specification: A roadmap,” in *Conference on The Future of Software Engineering (ICSE 2000)*. Limerick, Ireland: ACM, 2000, pp. 147–159.
- [65] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*. Print-Hall of India, 2002, no. 2.
- [66] R. Sammi, I. Rubab, and M. A. Qureshi, “Formal specification languages for real-time systems,” in *International Symposium on Information Technology*. Kuala Lumpur, Malaysia: IEEE, 2010, pp. 1642–1647.
- [67] ISO/IEC13568:2002, “Information technology — z formal specification notation — syntax, type system and semantics,” ISO/IEC/IEEE, Tech. Rep., 2002.
- [68] Y. Y. Haimes, “Modeling complex systems of systems with phantom system models,” *Systems Engineering*, vol. 15, no. 3, pp. 333–346, 2012.
- [69] H. J. Goldsby, P. Sawyer, N. Bencomo, B. H. C. Cheng, and D. Hughes, “Goal-based modeling of dynamically adaptive system requirements,” in *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2008)*. Washington, USA: IEEE Computer Society, 2008, pp. 36–45.
- [70] A. Piccolo, V. Galdi, F. Senesi, and R. Malangone, “Use of formal languages to represent the ertms/etcs system requirements specifications,” in *International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion, and, Road Vehicles (ESARS 2015)*. Aachen, Germany: IEEE, 2015, pp. 1–5.
- [71] RESPECT-IT, “A kaos tutorial,” Objectiver, Tech. Rep., 2007.
- [72] E. Yu, “i* an agent-and goal-oriented modelling framework,” Available at <http://www.cs.toronto.edu/km/istar/>, 2011, [Online, Accessed: December 18, 2016].
- [73] N. Bendov, “Designing for adaptability and evolution in system of systems engineering: Characterization of sos d 4.1,” INRIA, Germany, Tech. Rep., 2009.
- [74] E. Honour, “Verification and validation issues in systems-of-systems,” in *1st Workshop on Advances in Systems of Systems (AiSoS 2013)*. Roma, Italy: Larsen, Legay, Nyman (Eds.), 2013, pp. 2–7.
- [75] V. V. Graciano Neto, M. Guessi, L. B. R. Oliveira, F. Oquendo, and E. Y. Nakagawa, “Investigating the model-driven development for systems-of-systems,” in *European Conference on Software Architecture Workshops (ECSAW 2014)*. Vienna, Austria: ACM, 2014, pp. 1–8.
- [76] B. P. Zeigler and J. J. Nutaro, “Towards a framework for more robust validation and verification of simulation models for systems of systems,” *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 13, no. 1, pp. 3–16, 2015.
- [77] L. G. Wagner, K. S. R. Cofer, D. Collins, C. Rapids, and Iowa, “Formal methods tools qualification,” NASA, USA, Tech. Rep., 2017.
- [78] D. W. Cordes and D. L. Carver, “Generating a requirements specifications knowledge-base,” in *ACM Sixteenth Annual Conference on Computer Science (CSC 1988)*. Atlanta, Georgia, USA: ACM, 1988, pp. 727–.
- [79] T. H. Nguyen, M. Vo, B. Q. and Lumpe, and J. Grundy, “KBRE: a framework for knowledge-based requirements engineering,” *Software Quality Journal*, vol. 22, no. 1, pp. 87–119, 2014.
- [80] G. Abdalla, C. D. N. Damasceno, and E. Y. Nakagawa, “A systematic literature review on systems-of-systems knowledge representation,” Institute of Mathematics and Computational Sciences, University of São Paulo, São Carlos, Tech. Rep., 2015.
- [81] S. Luna, A. Lopes, H. Yan, S. Tao, F. Zapata, and R. Pineda, “Integration, verification, validation, test, and evaluation (IVT&E) framework for system-of-systems (SoS),” *Procedia Computer Science*, vol. 20, pp. 298–305, 2013.
- [82] J. Dahmann, J. Lane, G. Rebovich, and R. Lowry, “Systems of systems test and evaluation challenges,” in *5th International Conference on System of Systems Engineering (SoSE 2010)*. Loughborough, UK: IEEE, 2010, pp. 1–6.
- [83] IEC/TR 80002, “Medical device software - part 1: Guidance on the application of ISO 14971 to medical device software,” Association for the Advancement of Medical Instrumentation, USA, Tech. Rep., 2009.
- [84] EN 50126, “The specification and demonstration of reliability, availability, maintainability and safety (rams),” European Committee for Electrotechnical Standardization, Tech. Rep., 1999.
- [85] EN 50128, “Railway applications - communication, signalling and processing systems - software for railway control and protection systems,” European Committee for Electrotechnical Standardization, Tech. Rep., 2011.