Systems-of-Systems Development: Initiatives, Trends, and Challenges

Cristiane A. Lana*, Nilton M. Souza*, Márcio E. Delamaro*, Elisa Y. Nakagawa*, Flávio Oquendo†, José C. Maldonado*
*Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, SP, Brazil.
†IRISA/Université de Bretagne Sud - UBS, Vannes, France.
{cristiane.lana, niltonmendes}@usp.br, flavio.oquendo@univ-ubs.fr, {delamaro, elisa, jcmaldon}@icmc.usp.br

***Abstract*—Systems-of-Systems (SoS) refer to large, complex, and software-intensive systems, resulted from the interoperability among heterogeneous, independent constituent systems. The main purpose of SoS is to perform tasks that could not be achieved by these constituents separately; besides, unique SoS characteristics impose new challenges to their development processes. We conducted a Systematic Mapping to identify initiatives, trends, and challenges in the SoS development and, as a result, 32 initiatives were identified. We also evaluated these initiatives with regard to the adherence to the technical processes of IEEE/ISO/IEC 15288:2015 and to the software implementation essential processes of IEEE/ISO/IEC 12207:2008. In general, these initiatives emphasize artifacts in the early stages of the development processes, addressing mainly requirements and architecture. Moreover, 17 of them emphasize the testing activity. We also summarized the SoS characteristics addressed in the studies and discuss the main trends and challenges.**

***Index Terms*—Systems-of-Systems, Software Development Process, and Software Process Standards**

I. Introduction

Systems-of-Systems (SoS[1]) correspond to an class of complex and large software-intensive systems (e.g., for military, security, and aerospace), which result from the interoperability of several managerial and operationally independent software systems called constituents. Constituents are often developed by distinct companies, with their own stakeholders, development teams, processes, and resources [1, 2]. In addition, the development of a SoS often happens in an unknown environment and surrounded by uncertainties [2]. Due to that, SoS pose new challenges for the software engineering community, such as how to specify, conceive, design, and develop SoS to exhibit such inherent characteristics, such as constituents' managerial and operational independence, emergent behaviours, and evolutionary development. In this context, it is fundamental to comprehend how the traditional software development could be adapted or evolved to support those new SoS requirements, what initiatives are being established to develop SoS, and how to materialize those initiatives [3, 4, 5].

Some initiatives have been already materialized by the SoS community. An example is architectural frameworks that have been developed to reduce the SoS modelling complexity by using multiple views that represent the system from different perspectives (e.g., operational, functional, and system) [6]. The United States Department of Defense Architecture Framework (DoDAF) [7] and the British Ministry of Defence Architecture Framework (MODAF) [8] are the main architecture frameworks used for supporting the modelling of the SoS [6]. Other initiatives are being produced, for instance, the Vee model that has emerged as a way to represent different systems, including SoS [9].

It should be observed that there is no consolidated standard nor well-established terminologies that would make more uniform the advancements in the development of these initiatives for SoS. SoS development initiatives address different sets of development activities with different nomenclatures that not always include the basic activities considered in traditional software development process models [3, 4]

Taking into account the relevance of having well-established SoS development processes to achieve high quality products, the main objective of this study is to present a panorama about the current State of the art on SoS development processes, considering different application domains. To provide such a panorama, we performed a Systematic Mapping (SM) [10], a technique for finding and summarizing available pieces of evidence on a particular research topic. In this SM, we analyze the software development processes and infrastructure used for their establishment. The SoS characteristics addressed in the studies, as well as trends and challenges raised by the authors are also summarized and discussed.

Aiming at making uniform the analysis and description of found processes and related activities concerning SoS systematic development initiatives, current international standards related to software processes and products were used to analyze how these initiatives are adherent to well-established standards. Specifically, we use the international standards IEEE/ISO/IEC 15288:2015 [11], which addresses the systems and software engineering - system life cycle

[1]In this article, the acronym SoS has no variations between plural or singular.

978-1-5090-1633-4/16/$31.00

processes and IEEE/ISO/IEC 12207:2008 [12] that is based on IEEE/ISO/IEC 15288:2015 and defines software life cycle processes.

The remainder of this paper is organized as follows. In Section II, SoS and SoS system engineering process model are presented. Section III presents the planning, conduction, and report of the SM. In Section IV, the threats to validity are described and in Section V, conclusions and future work are discussed.

II. Background

In this section, the main concepts and terminology relevant for the understanding of this paper are presented.

A. System-of-Systems

The initial concept of SoS comes from several sources, as discussed by Nielsen et al. [13], and has been used in several application domains, such as energy, telecommunications, healthcare, and military environment. However, there is no consensus regarding the definition of SoS [13, 14]. The majority of the authors consider that SoS emerge when a set of needs can be met through a combination of various constituents systems [15]. Each constituent can be independently operated and managed and effectively interacts with others for meeting together new needs that each one could not address alone [15, 16].

In response to the widening recognition of SoS combined with the lack of a shared agreement on an SoS definition, Maier [14] proposes a set of five SoS characteristics: (i) *Operational Independence* of the constituent systems that provide their own functionalities even when they do not cooperate with another one; (ii) *managerial independence* of the constituent systems, in which each constituent can be individually managed from other systems and from the SoS; (iii) *evolutionary development* of the SoS that can evolve as a response to changes related to the environment, to the constituent systems, or to their functions and purposes; (iv) *emergent behavior* that enables the SoS to provide new functionalities resulted from interactions among constituent systems; and (v) *geographical distribution* of the constituent systems, which are physically decoupled and only exchange information among them. New proposed characteristics are either variations of this set or attached to specific application domains [13].

Figure 1 shows the four types of SoS topologies that are identified based on the relationship among the constituents in the SoS and on the levels of responsibility and authority [3]. SoS may be considered [3, 14]: (i) *virtual*, there is no central authority to manage activities and a clear SoS purpose; (ii) *collaborative*, the constituent systems work together more or less voluntarily to fulfill agreed upon central purposes, there is no SoS engineering team to guide or manage SoS related activities of the constituent systems; (iii) *directed*, it is centrally managed by a government, corporation, or Lead System Integrator (LSI) team and built to fulfill specific purposes. The constituent systems maintain their ability to operate independently, but evolution is predominantly controlled by SoS management organization; and (iv) *acknowledged*, when the objectives are organized, there is a designated manager, and resources allocated at the SoS level. But the SoS engineering team does not have complete authority over constituent systems. The constituent systems maintain their independent ownership, objectives, funding, and development approaches;

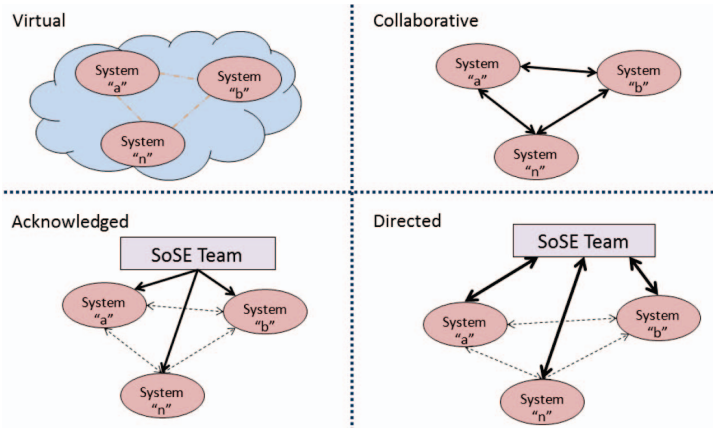

Figure 1. SoS Topologies [17]

The SoS characteristics and the diversity of constituent systems impose new challenges and problems that have been faced by the software and systems engineering areas in SoS development, as those discussed by the INCOSE SoS Working Group [18, 19]: (i) SoS Authorities; (ii) leadership; (iii) constituent systems' perspectives; (iv) capabilities and requirement; (v) autonomy, interdependence and emergence; and (vi) testing, validation, and learning.

B. System Engineering Process Models for SoS

The objective of the System Engineering Process Models (SEPM) is to support the use of system engineering (SE) to transform and evolve user's abstract operational need(s) into a physical system design solution that represents the optimal balance of technical, technology, cost, schedule, support solutions, and risks [20]. Organizations, specially in the military domains, such as the United States Army, United States Air Force (USAF), Department of Defense (DoD), International Council of System Engineers (INCOSE), and the Institute of Electrical and Electronics Engineers (IEEE), developed a variety of SEPM that have contributed to advance the state of the SE practice [21].

Among these SEPM, Vee model has emerged as a way of representing SE [9]. Since it was first developed in the 1980s, Vee model has been refined and applied in different domains, including in SoS [21]. Because of the particularities of SoS (see, Subsection II-A), a subarea called SoS Engineering or simply SoSE has emerged. SoSE has its own processes, methods and, techniques and the process used to conceive systems are structured in a small well-defined set of few large steps, which may be conducted by members of single

or multiple SoSE teams, depending on the size or scope of the SoS. SoSE process is composed by seven steps [22]: (i) Translating capability objectives; (ii) Understanding systems and relationships; (iii) Assessing performance to capability objectives; (iv) Developing and evolving SoS architecture; (v) Monitoring and assessing changes; (vi) Addressing requirements and solution options; and (vii) Orchestrating upgrades to SoS.

To address some key challenges of SoS development (e.g., interoperability), besides the aforementioned groups, new groups have emerged, including: (i) System-of-Systems Engineering Network (SoSEN); (ii) IEEE Technical Committee on Systems of Systems; (iii) International Consortium in Systems of Systems Engineering (ICSoSE); and (iv) United State System of Systems Engineering Network (US-SoSEN). Each group has specific purposes, including encompassing the study of strategies to research SoS; creating infrastructure (i.e., frameworks, tools, and methods) for developing SoS; increasing mainly the domain experts who are able to contribute to the industry; and seeking ways to mitigate the problem of interoperability [23].

III. Systematic Mapping

Our mapping was conducted in the context of software development processes for SoS. This mapping was carried out from April/2015 to October/2015 and involved six researchers, among them four experts. We used the guidelines established by Kitchenham and Charters [24] that is divided in three phases: planning, conducting, and reporting, as showed in Figure 2.

A. Phase 1: Planning

In this phase, the SM protocol was established, including the research questions, search strategy, and selection criteria. The SM protocol was elaborated by the author and validated by the experts.

1) Research Question:

In order to structure and simplify the research questions development, we used the PICO model [24], which includes the consideration of four elements: Population, Intervention, Comparison and Outcomes. *Population* comprises the developers and researchers of SoS and systems engineering community. *Intervention* for our SM is related to the context of SoS development processes. *Comparison* it not applicable. Lastly, in *Outcomes* characterize all primary studies identified in the literature that report SoS development processes. The research questions are:

RQ-01 Have SoS been constructed based on development processes? Which are those processes?
This RQ aims at identifying processes, but not excluding other initiatives (e.g., methods, frameworks, and models) proposed for building SoS.

RQ-02 Which SoS characteristics have been considered in SoS development processes?
We aim at investigate the main characteristics mentioned by the authors based on the five Maier's characteristics [14].

RQ-03 Which main challenges and trends have been discussed by the authors of the primary studies?
This RQ aims at identifying the main challenges and trends of SoS development processes discussed by the authors of the primary studies.

2) Search String: The search string was designed to cover variations and synonyms for terms related to "development process" and "systems-of-systems" and adapted to meet particularities of each search engine. To define the search string, five steps were taken: (i) search by synonyms of "development process" in systematic literature reviews of traditional software development processes; (ii) search in dictionaries; (iii) consulting experts; (iv) pilot search; and (v) use of control studies [4, 25, 26, 27]. The search string is:

("development process" OR "development method" OR "development approach" OR "development design" OR "design flow" OR "development procedure" OR "development life cycle" OR "agile development" OR "software product line") AND ("system of systems" OR "system-of-systems" OR "systems-of-systems" OR "systems of systems")

3) Search strategy:

We selected the following search engines: ACM Digital[2], IEEE Xplorer[3], ISI Web of Science[4], Science Direct[5], and Scopus[6]. Furthermore, snowballing was used as described by Wholin [28]. It was also considered the experts' opinion, gray literature, and related works of the included studies. In order to accordingly select the studies to answer our research questions, we established the following Inclusion (IC) and Exclusion Criteria (EC):

4) Inclusion criteria:

IC.1: The study discusses SoS development process.
IC.2: The study addresses SoS characteristics, problems or activities related to SoS development processes.

5) Exclusion criteria:

EC.1: The study is not related to SoS.
EC.2: The study does not discuss any SoS development process.
EC.3: Development process is not the main focus of the study.
EC.4: The study is an editorial, keynote, opinion, tutorial, poster or panel.
EC.5: The study is duplicated or there is newer or a more complete one about the same research.
EC.6: The study compiles results of other studies.
EC.7: The study is not written in English.
EC.8: The full text of the study is not available.

[2]http://dl.acm.org/
[3]http://ieeexplore.org/
[4]http://webofscience.com/
[5]http://www.sciencedirect.com/
[6]http://www.scopus.com/

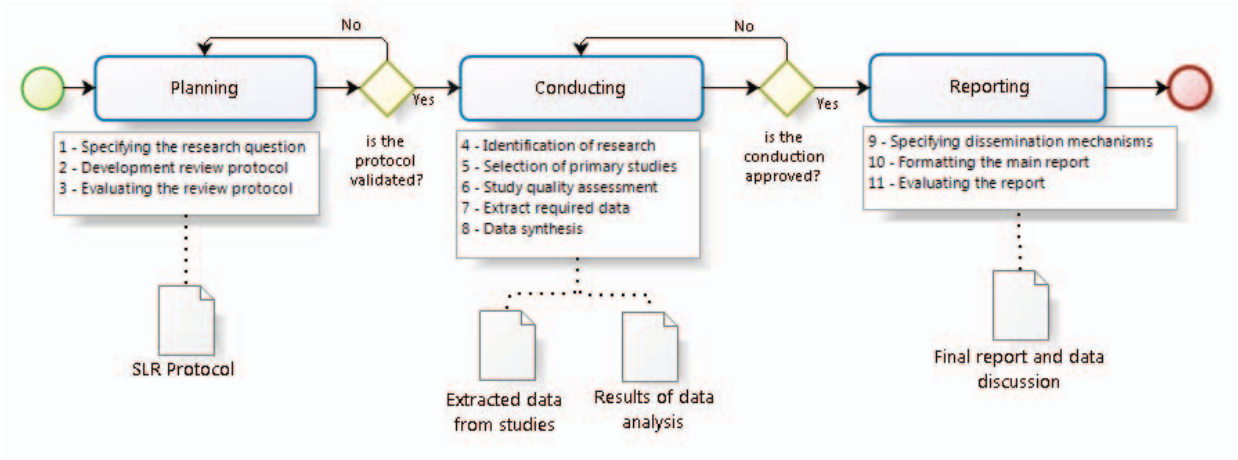

Figure 2. SM process based on the Kitchenham and Charters process [24]

B. Phase 2: Conduction

The identification and selection of the studies were executed according to the protocol established. We used the Start Tool[7] [29] to support the conduction process, organization and analysis of the primary studies in the first selection phase. In the second phase, we used the Microsoft Excel to organize and analyse the data. 493 primary studies were obtained considering all source of studies (i.e., engine search, snowballing, related works, and experts' opinion). After the application of the selection criteria using the title and abstract (i.e., first phase), 88 primary studies were selected for full reading and the selection criteria was applied again (i.e., second phase), resulting in 33 studies included. Out of the 33 studies, we considered just 32, since one of them (i.e., S27) in fact addresses software development infrastructure.

The 32 studies included are listed in Table I. Besides the studies, the table presents the identification (ID) and the publication year of each study. Out of these 32 primary studies, 65.63% of the studies (i.e., 21 studies) were published in the last five years and 90.63% ones (i.e., 30 studies) in the last ten years, what points to a growing interest, in the last decade, in the research area addressed in this paper.

Table I
List of Included Primary Studies.

Study ID	Included Study	Year
S1	Rossak et al. [30]	1994
S2	Ministry of Defence [8]	2005
S3	Mostermanm et al. [31]	2005
S4	Boehm et al.[32]	2006
S5	Shams et al. [33]	2008
S6	Gokhale et al. [34]	2008
S7	Carbon et al. [25]	2008
S8	Butterfield et al. [35]	2008
S9	Reichle et al. [36]	2008
S10	Kewley et al. [37]	2009
S11	Ballagny et al. [38]	2009
S12	Farcas et al. [39]	2010
S13	Decher [40]	2010
S14	Calinescu et al. [4]	2010
S15	Serugendo et al. [41]	2010
S16	Department of Defense [7]	2010
S17	Tu et al. [42]	2011
S18	Farroha et al. [26]	2011
S19	Dahmann et al. [3]	2011
S20	Mazzuchi et al. [43]	2011
S21	Mensing et al. [44]	2012
S22	Li and Yang [45]	2012
S23	Hallsteinsen et al. [5]	2012
S24	Santamaria et al. [46]	2012
S25	Ramos et al. [27]	2012
S26	Braga et al. [47]	2012
S28	Acheson et al. [48]	2013
S29	Luna et al. [49]	2013
S30	Madni and Sievers [50]	2014
S31	Louali et al. [51]	2014
S32	Sanduka et al. [6]	2014
S33	Ergin et al. [52]	2015

C. Phase 3: Reporting

Based on the analysis of the 32 primary studies included, we addressed the RQs.

1) RQ-01: Have SoS been constructed based on development processes? Which are those processes?:

We identified 32 initiatives of SoS development, as shown in Table II. Four of these initiatives are classified by the authors as processes, three as approaches, ten as frameworks, two as methods, and eight as models. For the other studies, we created the classification *Other Initiatives to SoS Development* that aggregates a single initiative of systems engineering, methodology, reengineering, integration ontology, and simulation platform established for SoS development.

Table III provides an overview of the initiatives regarding the domain type (i.e., generic, specific, or non specify),

[7]http://lapes.dc.ufscar.br/tools/start tool

Table II
Relation of proposed initiatives for SoS Development.

SoS Development Processes	ID
Service Driven Development Process (SDDP)	S5
SoSE Process	S8
Systems Engineering Process	S10
Model-based Development Process for Rich Services.	S12
SoS Development Approaches	**ID**
Agile Enterprise Systems Engineering approach	S18
Rule-Driven Methodology for Developing IT Ecosystems	S21
Product Line on Critical Embedded Systems (ProLiCES) Approach	S26
SoS Development Frameworks	**ID**
Generic Systems Integration Framework (GenSIF)	S1
British Ministry of Defence Architecture Framework (MODAF)	S2
Comprehensive Context Modeling Framework	S9
Framework for System-of-Systems Development	S14
Department of Defense Architecture Framework (DoDAF)	S16
Harmonized and Reversible Development Framework	S17
Framework of SPL-to-srSOS	S22
MUSIC Development Framework and Methodology	S23
Software integration Framework	S24
SoS Integration, Verification, Validation, Test, and Evaluation (IVVT&S) Framework	S29

(a)

SoS Development Methods	ID
MetaSelf Software Architecture and Development Method	S15
Architecture Optimization Method	S32
SoS Development Models	**ID**
Model-Based Design	S3
Scalable Spiral Process Model	S4
Component Synthesis using Model Integrated Computing (CoSMIC)	S6
Model of Components for Adaptive Systems (MOCAS)	S11
Time-sequenced Wave Model to SoS SE	S19
SOSE Integration Model	S20
Agent-Based Model	S28
Incentive-Based Negotiation Model	S33
Others Initiatives to SoS Development	**ID**
Collaborative Product Line Software and Systems Engineering (CoPulSE)	S7
Model-Based Design Methodologies and Supporting Design Tools	S13
Legacy Systems Reengineering to SPL	S25
SoS Integration Ontology	S30
System-level Simulation Platform for the Development of Unmanned Aircraft Systems - UAS	S31

(b)

application domains, activities adopted or attributes produced, and the type of SoS topology. A total of 75% of the initiatives were developed to be applied to specific domains, while 12.50% (S4, S17, S25, and S28) did not specify the application domain and 12.50% of them (S1, S2, S11, and S16) are classified as generic applications, i.e., they would be applied to any application domain. Although initiatives S2, S11, and S16 were classified as generic, their development were realized inside specific domains: S2 and S16 were proposed in the defense domain and S11 in the self-adaptation systems domain. In the group of specific domains, the most addressed ones, with five studies, is defense in the military scope, followed by Unmanned Aircraft Systems, with four studies, and avionics, with two studies, in the military and civil scope. Other application domains were addressed by only one initiative, as presented in Table III.

The main artifacts discussed by the included studies are software architecture, with 96.88% of them, and software requirements, with 93.75%, providing evidences that SoS development community are addressing relevant artifacts in the early stages of the development. Regarding the development activities, Verification, Validation, and Test (VV&T) were addressed in 71.88% of the initiatives, followed by system design in 53.13% of the initiatives and component design in 9.38% ones. Although VV&T have been addressed in a high percentage of the studies, they are, in most of the cases, just mentioned and treated superficially. This fact raises a read flag concerning the quality of delivered software.

Regarding topologies, only seven studies (S5, S10, S19, S22, S25, S28, and S33) were developed for specific topologies. The most used topology was acknowledged, with five occurrences, followed by directed, with four occurrences, then collaborative, with two, and virtual, with one occurrence. It is noticed that a same initiative can be applied to more than one topology type. For instance, the initiative developed in S5 can be applied to virtual, collaborative and directed topologies. S25 can be applied to collaborative, directed, and acknowledged and S22 to directed and acknowledge. Similarly, studies S12 and S23 were established for two application domains. S12 encompassed avionics and automotive domains and S23 addressed mobile and ubiquitous computing one.

After the overview of the initiatives, we evaluated each one with regard to the coverage of the technical processes of the standard IEEE/ISO/IEC 15288:2015 [11] and the software implementation essential processes of the standard IEEE/ISO/IEC 12207:2008 [12]. In the analysis of the adhesion to the standards, each initiative was evaluated considering activities that were explicitly described by the authors of the included studies.

Table IV depicts the initiatives in decreasing order of adherence to the standards, the percentage of adherence of these initiatives for each standard and full coverage of each technical or essential process. It should be highlighted that, a specific process has been addressed in a specific initiative does not mean that the initiative covers all the process activities defined by the standards. It just means that the specific process has been explicitly mentioned in the initiative.

Considering the standard ISO/IEC/IEEE 15288:2015, the initiative proposed in S20 had the highest adherence, 71.14%, followed by S13, S19, S26, and S28, all of them with 64.71% of adherent to the standard. Regarding the technical processes, the *Business or Mission Analysis Process* was fulfilled by all the initiatives, followed by *Stakeholders needs and Requirements Definition* and *Architecture Definition*

Table III
Characterization of the included studies

			Classification performed by the authors of the included studies																																				
			Processes				Frameworks										Methods		Models								Approaches			Other Initiatives										
			S5	S8	S10	S12	S1	S2	S9	S14	S16	S17	S22	S23	S24	S29	S15	S32	S3	S4	S6	S11	S19	S20	S28	S33	S18	S21	S26	S7	S13	S25	S30	S31	Total	%
Domains	Type	Generic					x	x			x											x													4	12.5
		Specific	x	x	x	x			x	x			x	x	x	x	x	x	x		x		x	x		x	x	x	x	x		x	x	24	75.00
		No specify										x								x						x						x			4	12.5
Application Domains	Military	Defense			x			x			x							x					x			x	x								7	21.87
		Large Scale Systems																						x												1	3.12
	Military and Civil	Ultra Large Scale Systems	x																																	1	3.12
		Real-Time and Embedded System																			x															1	3.12
		GEOSS		x																																1	3.12
		Pervasive Computing Systems							x																											1	3.12
		Avionics				x																									x					2	6.25
		Automotive Systems				x																														1	3.12
		Large Scale Systems								x																										1	3.12
		Self-Organising Systems															x																			1	3.12
		Mobile Computing												x																						1	3.12
		Ubiquitous Computing												x																						1	3.12
		Unmanned Aircraft Systems													x	x													x						x	4	12.5
	Civil	Automotive Systems																	x																	1	3.12
		Office																												x						1	3.12
		Self-Adaptation Systems																				x														1	3.12
		IT Ecosystems																										x								1	3.12
		Airport Systems											x																							1	3.12
		Earth Seismic Studies																															x			1	3.12
Artifacts/Activities	Type	Software Architecture	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x		31	96.87
		Software Requirements	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x	x	x	x			30	93.75
		VV&T	x	x		x	x			x	x	x		x	x	x	x	x			x		x	x			x	x	x	x	x	x	x	x		23	71.87
		Component Design		x		x							x																								3	9.37
		System Design		x	x	x	x			x	x	x		x	x								x	x	x		x	x	x		x				x		17	53.12
Topology SoS	Type	Virtual	x																																	1	3.12
		Collaborative	x																													x				2	6.25
		Directed	x		x								x																			x				4	12.5
		Acknowledged											x										x		x	x						x				5	15.62

Processes that achieved the same coverage: 90.63%. The *Validation Process* was covered by 56.25% and *Implementation Process* by 50%. The other technical processes did not achieve at least 50% of coverage. However, the adherence of the initiatives could be related with the type of application it addresses. As can be seen in Table III, most of the initiatives were developed for specific applications/domains, in this case, the set of chosen activities may be related to the most relevant ones to the target domain. The authors of the studies did not make explicit their decisions.

It was also evaluated the adherence of each initiative in relation to the processes considered essential for the implementation of software, described in the standard IEEE/ISO/IEC 12207:2008. In this analysis, initiatives were evaluated against the following six processes: (i) software requirements analysis process; (ii) software architectural design process; (iii) software detailed design process; (iv) software construction process; (v) software integration process; and (vi) software qualification testing process.

It should be understood in advance that our analysis was based on explicitly mentioned information/process by the authors of the studies. It should also be understood that although initiatives S11, S16, S18, S25, S31, and S32 discuss some main activities and/or artifacts of software development processes, as illustrated in Table III, they did not address these activities and/or artifacts in building their initiatives propositions. For example, S11 and S31 discuss software architecture, but do not mention explicit *Software Architectural Design Process*. S25 and S32 discuss VV&T, but they do not explicitly address *Software Qualification Testing Process*. S16 and S18 highlight the importance of systems design, but they do not explicitly establish the *Software Detailed Design Process*.

The *Software Requirements Analysis* and *Software Architectural Design* Processes remain the most covered by the initiatives, with 90.63%. The next most covered was the *Software Qualification Testing Process* with 65.63%. This standard (i.e., IEEE/ISO/IEC 12207:2008) does not distinguish among the activities of verification, validation and testing. Thus, these activities are included under this process. The *Software Construction Process* achieved a coverage of 50% of the initiatives, followed by the *Software Detailed Design Process* and *Software Integration Process*, with 46.88% of coverage.

It should be noticed that the initiatives reported in S13, S14, S19, S20, S24, S26, and S29 achieved 100% of adherence to the standard IEEE/ISO/IEC 12207:2008, but

Table IV
ADHERENCE ANALYSIS OF THE INCLUDED STUDIES

		Classification performed by the authors of the included studies																																										
		Processes					Frameworks											Methods		Models									Approaches			Others Initiatives														
ISO/IEC/IEEE 15288:2015	**Technical Processes**	S12	S5	S8	S10	S24	S14	S23	S17	S29	S1	S22	S9	S16	S2	S15	S32	S20	S19	S28	S4	S6	S3	S11	S33	S26	S21	S18	S13	S7	S30	S25	S31	Total	%											
	Business or Mission Analysis	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	32	100.00											
	Stakeholders needs and Requirements Definition Process	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X		29	90.63											
	System Requirements Definition Process				X											X		X	X	X						X								6	18.75											
	Architecture Definition Process	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X			29	90.63											
	Design Definition Process	X		X	X	X	X	X	X		X							X	X	X						X	X	X	X				X	16	50.00											
	System Analysis Process																																	0	0.00											
	Implementation Process	X	X			X	X	X	X		X					X		X	X	X	X	X				X	X		X					16	50.00											
	Integration Process	X	X		X	X	X											X	X	X			X			X		X	X	X	X	X		15	46.88											
	Verification Process	X		X		X	X			X						X		X			X			X		X	X		X					12	37.50											
	Transition Process		X															X											X					3	9.38											
	Validation Process	X	X	X		X		X	X	X								X	X	X	X	X				X	X		X	X	X		X	18	56.25											
	Operation Process							X														X												2	6.25											
	Maintenance Process																		X	X	X													3	9.38											
	Disposal Process																																	0	0.00											
	Total adherence	8	7	6	6	8	7	7	6	5	5	3	3	3	3	6	3	10	9	9	7	6	3	3	2	9	7	5	9	5	5	3	3											
	%	57.14	50.00	42.86	42.86	57.14	50.00	50.00	42.86	35.71	35.71	21.43	21.43	21.43	21.43	42.86	21.43	71.14	64.29	64.29	50.00	42.86	21.43	21.43	14.29	64.29	50.00	35.71	64.29	35.71	35.71	21.43	21.43											
ISO/IEC/IEEE 12207:2008	**Software Implementation Essential Processes**	S12	S5	S8	S10	S24	S14	S23	S17	S29	S1	S22	S9	S16	S2	S15	S32	S20	S19	S28	S4	S6	S3	S11	S33	S26	S21	S18	S13	S7	S30	S25	S31		%											
	Software Requirements Analysis P	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X		29	90.63											
	Software Architectural Design Pro	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X			29	90.63											
	Software Detailed Design Process	X		X	X	X	X	X	X		X							X	X	X						X	X		X				X	15	46.88											
	Software Construction Process	X	X			X	X	X	X		X					X		X	X	X	X	X				X	X		X					16	50.00											
	Software Integration Process	X	X		X	X	X											X	X	X			X			X		X	X	X	X	X		15	46.88											
	Software Qualification Testing Pro	X	X	X		X	X	X	X	X						X		X	X	X	X	X		X		X	X		X	X	X		X	21	65.63											
	Total adherence	6	5	4	4	6	6	5	5	3	4	2	2	2	2	4	2	6	6	6	4	4	2	2	1	6	5	3	6	4	4	2	2											
	%	100.00	83.33	66.67	66.67	100.00	100.00	83.33	83.33	50.00	66.67	33.33	33.33	33.33	33.33	66.67	33.33	100.00	100.00	100.00	66.67	66.67	33.33	33.33	16.67	100.00	83.33	50.00	100.00	66.67	66.67	33.33	33.33											

they are not classified as software development processes by their authors. However, they mention all essential processes. In this way, these initiatives are classified as software development processes[8]. On the order hand, initiatives classified as software development processes by the initiatives' authors, for instance, S5, S8, and S10, do not explicitly address all the essential processes. The initiative presented in S5 does not address the *Software Detailed Design Process* and S10 the *Software Construction* and *Software Qualification Testing* Processes. Although the initiative presented in S8 is based on SoSE Process, which is adherent to the standard IEEE/ISO/IEC 12207:2008, this initiative does not explicitly address the *Software Construction Process* and *Software Integration* Processes.

It is interesting to note that many initiatives, such as S1, S2, S8, S9, S15, S16, S17, S21, S22, S23, S29, and S32, do not address the software integration essential process, which, from our point of view, should be mandatory. Specifically, S17, S21, and S23 in fact address all essential process, but not the software integration one.

It should also be highlighted that some of the initiatives did not intend from the beginning to cover all the essential processes, such as the initiatives S2, S7, S16, S25, S29, and S32, since they have specific purpose inside the software development process. For instance, S2, S7, and S16 focus on the development of SoS architectures and fulfills on average 33.33% of the essential processes. S25 addresses legacy systems reengineering for software product line and also address 33.33% of the essential processes. The initiative presented in S29 is focused on the Integration, Verification, Validation, Testing, and Evaluation (IVVT&E) of SoS with 50% of adherence, and in S32, the focus is in simulation platforms with 33.33% of adherence.

2) RQ-02: Which SoS characteristics have been considered in SoS development processes?:

As described in Section II-A, SoS have specific characteristics that differentiate them from the constituent systems and somehow should be addressed in the software development processes. The included studies were looked at considering the set of characteristics defined by Maier [14].

Among Maier's characteristics, the most considered one in the studies is *independence of constituents*, with 65.63% of the studies (i.e., 21 studies). The independence may be managerial or operational, but in many of the studies, these two characteristics were refereed to only as *independence*, nomenclature adopted in Table Va.

[8]We considered the definitions of the standards IEEE/ISO/IEC 12207:2008 and ISO/IEC/IEEE 15288:2015.

The second characteristic is *emergent behavior* that was considered by 46.88% of the studies (i.e., 15 studies). The characteristics *evolutionary development* (or simply evolutionary) is discussed in only in 25.00% of them (i.e., 8 studies). Table V summarizes all characteristics found in the included studies. Maier's characteristic are showed in Table Va, which presents the year of the first occurrence of the characteristics. For instance, the characteristics *distributed* is found the first time in S1 published in 1994, even four years before being mentioned by Maier. This shows that the SoS characteristics had been considered and discussed even before their consolidation and clustering in 1998 by Maier.

Besides Maier's characteristics, eight additional characteristics are observed in the included studies, as shown in Table Vb. In this group, the characteristics *reconfigurable* and *common mission* (or simply *mission*) are considered by 34.38% of the studies (i.e., each characteristic was considered by 10 studies). The characteristic less considered is *self-adaptation*, discussed only by 9.09% of them (i.e., three studies).

Adaptability, *complexity*, and *interoperability* were considered in some studies as characteristics of SoS. However, according to the standard ISO/IEC 25010:2011 [53], which addresses "System and Software Quality Requirements and Ealuation (SQuaRE)", *adaptability* and *interoperability* are system product quality characteristics (i.e., non-functional requirements). Moreover, in the systematic literature review performed by Bianchi et al. [2], *complexity* is understood as a quality characteristics not covered by the standard ISO/IEC 25010:2011. Table Vc exhibits these quality characteristics, the studies that address them, and the year of their first occurrence, considering the included studies. *Interoperability* was considered by 46.88% of the studies (i.e., 15 studies). According to the standard ISO/IEC 25010:2011, *interoperability* is a sub-characteristics of the characteristic *compatibility*[9]. *Adaptability* is a sub-characteristics of *portability*[10] and was addressed in 28.13% of the studies (i.e., nine studies). Lastly, the *complexity* it was considered in only 16.63% of the studies (i.e., five studies).

In general, we observed that only S30 explicitly reported all Maier's characteristics of SoS and a total of 81.25% of the studies (i.e., 26 studies) considered at least one of these characteristic. Others 9.09% addressed only characteristics that compose the group of "Other characteristics observed" or "quality characteristics". Studies S2 and S3 did not refer to none of them.

Likewise, the type of application also impacts the choice or prioritization of characteristics and quality characteristics, not only the choices of processes that compose the set of SoS development initiatives. For instance, the study S6, which discusses the problems of proprietary solutions to the Distributed Real-time and Embedded Systems, considered the characteristics *emergent behavior*, *heterogeneous*, and *interoperability*. It is important to say that the authors did not provide information on the reason to choose these characteristics in their work.

3) RQ-03: Which main challenges and trends have been discussed by the authors of the primary studies?:

The main challenges and trends discussed by the authors of the included studies about SoS development are identified and described in this section.

According to Boehm [54, 40], in terms of systems and software development precesses for SoS, the fact that the capability of these SoS to be emergent rather than specifiable prespecifiable has become the primary challenge, impacting the complexity of the development process. Yet according to Boehm, emergence of requirements is incompatible with past practices such as requirements driven sequential, waterfall process models, formal programming calculi, and with process maturity models. To mitigate and manage this new complexity, more adaptive and risk-driven system and software development process should be in place [54]. Moreover, engineers must have more available detailed and comprehensive infrastructure available, i.e., systems engineering processes, tools, methods, technologies, and standards for SoS. The use of these new infrastructure will enable engineers to more efficiently and effectively develop a balanced SoS's solution [35].

Traditional software development process models (or simply traditional models) have consolidated knowledge and have been evaluated for many years. These traditional models can constitute a basis for new models, for example the M-model presented in S20. The phases of development in these models were and still are extensively researched and successful experiences already achieved can serve as a basis to develop or even evolve such new infrastructure (e.g., tools, methods, techniques, models, and standards) to be applied in SoS development. The challenge is to identify each real contribution of the traditional models, as well as their infrastructure, and dispose non-relevant information. Furthermore, traditional models can support a better understanding of the development of SoS constituents [3, 4, 5, 17].

In the SoS development scope, two approaches widely used are Model-Based Systems Engineering (MBSE) and Service-Oriented Architecture (SOA). Techniques of MBSE were adopted by several studies (i.e., 59.38% of the included studies) for managing systems complexity by enabling engineers to better understand requirements, develop candidate architectures, and verify design decisions early in the development process [40]. An overview of the benefits of using MBSE techniques can be seen in Nielsen et al. [13], which discussed several challenges and trends related to the SoS modelling, architectural description, simulation,

[9]Compatibility is the degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment

[10]Portability is the degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another

Table V
SoS Characteristics

Maier's Characteristics	ID	%	First Occurrence
Emergent Behavior	S4, S5, S6, S8, S10, S14, S15, S18, S19, S20, S24, S26, S29, S30, S33	46.88	2006
Evolutionary	S4, S5, S20, S21, S25, S29, S30, S32	25.00	2006
Distributed	S1, S5, S8, S12, S17, S20, S23, S29, S30, S32	31.25	1994
Independence	S1, S4, S5, S8, S11, S12, S14, S15, S16, S18, S19, S20, S21, S22, S23, S25, S28, S29, S30, S32, S33	65.63	1994

(a) SoS characteristics found in the included studies

Other Characteristics Observed	ID	%	First Occurrence
Autonomy	S11, S14, S15, S20, S21, S22, S30, S31	25.00	2008
Collaboration	S7, S8, S14, S15, S22, S33	18.75	2008
Dynamic Architecture	S4, S7, S8, S29, S31	15.63	2006
Heterogeneous	S6, S14, S23, S30	12.50	2008
Interdependence	S14, S16, S19, S28, S29, S30	18.75	2008
Mission	S1, S8, S18, S19, S20, S21, S22, S25, S29, S30	31.25	1994
Reconfigurable	S8, S11, S13, S16, S20, S21, S23, S26, S29, S30.	31.25	2008
Self-Adaptation	S15, S19, S33	9.38	2010

(b) Other SoS characteristics found in the included studies

Quality Characteristics	ID	%	First Occurrence
Adaptability	S9, S11, S14, S15, S19, S23, S28, S30, S33	28.13	1994
Complexity	S12, S14, S21, S25, S32	15.63	2008
Interoperability	S1, S4, S6, S7, S8, S10, S14, S15, S16, S17, S18, S20, S22, S25, S30	46.88	1994

(c) Quality Characteristics considered by the included studies

verification, and testing. SOA was adopted for 21.88% of the initiatives to architect SoS and minimize interoperability and integration problems. Because of the features, such as platform independence, loose coupling, and support for security, SOA solutions represent strong candidates for implementing new computer systems or front-ends to legacy systems that need to be integrated into a SoS [4]. However, inappropriate application of SOA-principles results in a high degree of fragmentation and scattering of functionality, leading to additional difficulties in requirements traceability and quality assurance of the SoS [39].

It is noteworthy that in traditional models, integration is usually considered in the late stages of the development, while in SoS context, it would be a problem to be treated in the early stages [39, 26], because the SoS formation occurs through the interoperability of their constituents. These constituents are typically designed without knowledge and with different technologies of the other constituents members. This diversity brings impact to the activities of integration [4, 50]. Thus, requirements elicitation and software architectures must support the system integration rigorously throughout the entire SoS development process and during reconfigurations. Challenges include the management of multiple versions of requirement specifications and the ability to verify consistency, beyond of the integration of heterogeneous and distributed constituents into a highly complex SoS with a wide variety of functional and non-functional requirements. [39]. Moreover, with regard to the architecture design, it is necessary to identify different suppliers of the constituents, adapting the architecture to meet evolving requirements and supplier limitations or constraints, overseeing the implementation efforts, planning and executing the SoS integration, and VV&T the SoS and its constituents [32].

In the context of IVVT&E, the research has focused on methodologies for SoS capabilities IVVT&E. The use of validation in SoS is necessary to assess if new SoS capabilities and/or implemented changes in the constituent systems meet the required SoS mission. In this sense, scalability of the VV&T activities for SoS is a major concern, in particular, when a large number of constituents are involved and IVVT&E may be too costly or time-consuming to be implemented within a limited period of time. Furthermore, Testing and evaluation must be prepared to handle both certain and uncertain test requirements and would require new tools and methods to address SoS in environments with uncertain operating scenarios [17].

The traditional testing infrastructure also needs to evolve to meet the testing of SoS, as discussed in S29. Typically, as the number of systems, interconnections, and interface protocols grow over time, the system's complexity increases and the SoS becomes difficult to maintain [17]. Additionally, the constituent systems can evolve asynchronously or participate in more than one SoS, impacting the integration process, the testing of these constituents and of the own SoS [17]. To ensure that SoS is not affected by the constituent system upgrades or changes, it is necessary to expand the constituent system-level regression testing to the SoS level [17].

Some challenges were discussed by studies in specific domains, as S6 and S14 that addressed challenges in Distributed Real-time and Embedded (DRE) systems, and Large Scale Systems, respectively. In these systems, the main challenges are in the proprietary hardware and software technologies and development techniques.

Unfortunately, proprietary solutions often fail to address the needs of large-scale DRE systems over their extended life cycle [34]. For instance, as DRE systems grow in size and complexity, the use of proprietary technologies can make it hard to adapt DRE systems to meet new functional or quality of service (QoS) requirements, hardware/software technology innovations, or emerging market opportunities [34].

In the large-scale systems domain, several software engineering techniques and paradigms, such as SOA, formal verification, policy-based autonomic computing, were examined by Calinescu and Kwiatkowska [4] to tackle some of the challenges associated with the SoS development. Figure 3 depicts the SoS challenges and the techniques or paradigms to minimize them. Eight techniques or paradigms can minimize eight SoS challenges. For instance, the adaptability challenge can be supported by model-driven development/code generation, dynamic reconfiguration, and on line machine learning. Dependability (assurance) and predictability involve formal verification. The description in details of each technique and paradigm can be found in [4].

	SoS challenges							
Techniques and paradigms	interoperability/security	dependability (assurance)	collaboration	global-objective specification	predictability	adaptability	longevity	flexibility
Service-oriented architectures	✓							
Policy-based autonomic computing				✓				✓
Formal Verification		✓			✓			
Model-driven development/code generation						✓		
Component-based development			✓					
Dynamic reconfiguration						✓	✓	✓
Online machine learning						✓	✓	
Resource discovery			✓					✓

Figure 3. Software engineering techniques and paradigms that can help address SoS challenges [4]

To summarize, current trends towards larger, complex software intensive-systems, and SoS require more complex, robust software engineering processes to better support technical processes of requirements, design, architecture, integration, and VV&T founded by systems engineering infrastructures, eventually inspired in traditional infrastructures [3, 32, 50]. Furthermore, some skills are necessary for the development of successful SoS, such as [32]: (i) achieve timely decisions with a potentially diverse set of stakeholders; (ii) quickly resolve conflicting needs; and (iii) coordinate activities of multiple suppliers who are currently working together to provide capabilities for a SoS, but are often competitors on other system development efforts.

IV. Threats to Validity

The threats to validity of the SM carried out in this paper and the actions taken to mitigate them are:

- **Missing of important primary studies:** This SM used a specific group of search engines considered as the most relevant ones according to [24, 55]. However, primary studies indexed by others search engines may be missing. To mitigate this threat, we adopted the snowballing, experts' opinion, and gray literature. In addition, we evaluated related works of the included studies.
- **Choice of primary studies:** Studies were selected based on the classification of the authors, restricted to SoS. No other nomenclature (e.g., autonomic computing, software ecosystem) was considered.
- **Number of reviewers:** SM was conducted by one researcher. Even considering the strategies adopted and the support of experts, the number of reviewers may imply risk of bias;
- **Study no available:** Ten primary studies retrieved by the search engines are not available, despite the efforts to contact the authors by email. These studies represented approximately 2.36% of the retrieved studies; and
- **Data Extraction:** Some information was not clearly available in the studies and had to be interpreted. In order to ensure the validity of our SM, discussions with experts were carried out whenever there was doubts.

V. Conclusion and Future Works

This study presents a relevant panorama about the current State of the art on SoS development processes as well as challenges and trends related to SoS development. SoS research area is relatively new and most of the publications are concentrated in the last ten years, total of 96.88% (i.e., 31 studies were published from 2005 to 2015).

Moreover, we identified, analyzed, and classified approaches of SoS development. We evaluated these approaches regarding the adherence to the technical processes of the standard IEEE/ISO/IEC 15288:2015 and the software implementation essential processes of the standard IEEE/ISO/IEC 12207:2008. These approaches emphasize the early stages of the development process, addressing requirements and architecture, besides of the activity of the testing area. We summarized the SoS characteristics addressed in the studies and discuss trends and challenges raised by the authors.

Many of the 32 initiatives are adherent to the above mentioned standards, but few evidence is provided on their effectiveness. The community still lacks of a better support to the technical processes of requirements, design, architecture, integration, and VV&T supported by systems engineering infrastructures, towards the achievement of high-quality SoS development.

It can be inferred from the given scenario that promoting integration among requirements elicitation, architecture design, and IVVT&E would promote the evaluation of corresponding artifacts in the early stages, of the SoS development processes, even improving these processes.

Our future work will concentrate in facilitating the validation of the SoS requirements, embodying validation information into architecture descriptions, providing facilities for planning and reusing this information inside SoS development processes.

VI. Acknowledgments

The authors thank the Software Architecture Group (START) of the ICMC/USP for its advice and guidance to improve this work and CAPES, CNPq, and FAPESP for the financial support of this work.

References

[1] S. Y. Han and D. DeLaurentis, "Development interdependency modeling for system-of-systems(sos) using bayesian networks: Sos management strategy planning," in *CSER*, 2013, pp. 698–707.

[2] T. Bianchi, D. S. Santos, and K. R. Felizardo, "Quality attributes of systems-of-systems: A systematic literature review," in *SESoS*, 2015, pp. 23–30.

[3] J. Dahmann, G. Rebovich, J. Lane, R. Lowry, and K. Baldwin, "An implementers' view of systems engineering for systems of systems," in *SysCon*, 2011, pp. 212–217.

[4] R. Calinescu and M. Kwiatkowska, "Software engineering techniques for the development of systems of systems," in *Monterey*, 2008, pp. 59–82.

[5] S. Hallsteinsen, K. Geihs, N. Paspallis, F. Eliassen, G. Horn, J. Lorenzo, A. Mamelli, and G. Papadopoulos, "A development framework and methodology for self-adapting applications in ubiquitous computing environments," *Journal of Systems and Software*, vol. 85, no. 12, pp. 2840–2859, 2012.

[6] I. Sanduka and R. Obermaisser, "Model-based development of systems-of-systems with real-time requirements," in *INDIN*, 2014, pp. 188–194.

[7] DoD Deputy Chief Information Officer, "United states department of defense architecture framework (DoDAF)," Department of Defense of United States, Technical Report Version 2.02, Tech. Rep., 2010.

[8] M. Partners, "British ministry of defense architecture framework (modaf): Technical handbook," MODAF Partners,Technical Report Version 1.0, Tech. Rep., 2005.

[9] E. Ryen, "Overview of the system engineering process," North Dakota Department of Transportation - NDDOT, Tech. Rep., 2008.

[10] K. Pertensen, 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.

[11] IEEE, "Systems and software engineering: Software life cycle processes - ISO/IEC/IEEE 15288:2015," Institute of Electrical and Electronics Engineers, Incorporated., Tech. Rep., 2015.

[12] ——, "Systems and software engineering: Software life cycle processes - ISO/IEC/IEEE 12207:2008," Institute of Electrical and Electronics Engineers, Incorporated, Tech. Rep., 2008.

[13] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "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.

[14] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998.

[15] D. A. DeLaurentis, "A taxonomy-based perspective for systems of systems design methods," in *SMC*, 2005, pp. 86–91.

[16] E. Y. Nakagawa, M. B. Gonçalves, M. Guessi, L. B. R. Oliveira, and F. Oquendo, "The state of the art and future perspectives in systems of systems software architectures," in *SESoS*, 2013, pp. 13–20.

[17] J. A. Lane, "What is a system-of-system and why should i care?" University of Southern California, Tech. Rep., 2013.

[18] J. Dahmann, "The state of systems of systems engineering knowledge sources," in *SoSE*, 2015, pp. 475–479.

[19] D. D. Walden, G. J. Roedler, K. J. Forberg, D. R. Hamelin, and T. M. Shortell, "INCOSE systems engineering handbook: a guide for systems life cycle processes ans activities," Version 3.2.2. 4ed., John Wiley & Sons Inc, Tech. Rep., 2015.

[20] C. S. Wasson, *System analysis, design, and development : concepts, principles, and practices*. John Wiley & Sons Inc, 2006, no. 1.

[21] ——, *System Engineering: analysis, design, and development*. John Wiley & Sons Inc, 2015, no. 2.

[22] K. Baldwin, "System engineering guide for systems-of-systems engineering," Department of Defense of United States, Tech. Rep., 2008.

[23] C. Ncube, "On the engineering of systems of systems: Key challenges for the requirements engineering community," in *RESS*, 2011, pp. 70–73.

[24] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University Joint Report, Tech. Rep., 2007.

[25] R. Carbon, G. Johann, D. Muthig, and M. Naab, "A method for collaborative development of systems of systems in the office domain," in *ECOC*, 2008, pp. 339–345.

[26] D. Farroha and B. Farroha, "Agile development for system of systems: cyber security integration into information repositories architecture," in *SysCon*,

2011, pp. 182–188.
[27] M. Ramos, P. Masiero, R. Braga, and R. Penteado, "Reengineering legacy systems towards system of systems development," in *IRI*, 2012, pp. 394–401.
[28] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *EASE*, 2014, pp. 38:1–38:10.
[29] S. Fabbri, E. M. Hernandes, A. Di Thommazo, A. Belgamo, A. Zamboni, and C. Silva, "Managing literature reviews information through visualization," in *ICEIS*, 2012, pp. 36–45.
[30] W. Rossak, T. Zemel, V. Kirova, and L. Jololian, "A two-level process model for integrated system development," in *ECBS*, 1994, pp. 90–96.
[31] P. J. Mostermanm, J. Ghidella, and J. Friedman, "Model-based design for system integration," in *CDEN*. Open Journal Systems, 2005, pp. 1–10.
[32] B. Boehm and J. Lane, "21 st century processes for acquiring 21 st century software-intensive systems of systems," *The Journal of Defense Software Engineering*, vol. 19, no. 5, pp. 4–9, 2006.
[33] F. Shams, A. Sharifloo, M. Mirakhorli, and M. Emaeli, "A service driven development process (SDDP) model for ultra large scale systems," in *ULSSIS*, 2008, pp. 37–40.
[34] A. Gokhale, K. Balasubramanian, A. Krishna, J. Balasubramanian, G. Edwards, G. Deng, E. Turkay, and J. Parsons, "Model driven middleware: a new paradigm for developing distributed real-time and embedded systems," *Science of Computer Programming*, vol. 73, no. 1, pp. 39–58, 2008.
[35] M. Butterfield, J. Pearlman, and S. Vickroy, "A system-of-systems engineering geoss: architectural approach," *IEEE Systems Journal*, vol. 2, no. 3, pp. 321–332, 2008.
[36] R. Reichle, M. Wagner, M. U. Khan, K. Geihs, J. Lorenzo, M. Valla, C. Fra, N. Paspallis, and G. A. Papadopoulos, "A comprehensive context modeling framework for pervasive computing systems," in *DAIS*, 2008, pp. 281–295.
[37] R. H. Kewley Jr. and A. Tolk, "A systems engineering process for development of federated simulations," in *SpringSim*, 2009, pp. 1–8.
[38] C. Ballagny, N. Hameurlain, and F. Barbier, "MOCAS: A state-based component model for self-adaptation," in *SASO*, 2009, pp. 206–215.
[39] C. Farcas, E. Farcas, I. Krueger, and M. Menarini, "Addressing the integration challenge for avionics and automotive systems: from components to rich services," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 562–583, 2010.
[40] P. Decher, "Requirements driven development from contract win to customer sign-off," in *Monterey*, 2010, pp. 1–8.
[41] D. M. G. Serugendo, J. Fitzgerald, and A. Romanovsky, "MetaSelf: An architecture and a development method for dependable self- systems," in *SAC*, 2010, pp. 457–461.
[42] Z. Tu, G. Zacharewicz, and D. Chen, "Harmonized and reversible development framework for hla based interoperable application," in *DEVS*, 2011, pp. 51–58.
[43] T. Mazzuchi and G. Albakri, "System of systems engineering facilitates integration of large scale complex systems," *INCOSE*, vol. 21, no. 1, pp. 811–855, 2011.
[44] B. Mensing, U. Goltz, A. Aniculaesei, S. Herold, A. Rausch, S. Gartner, and K. Schneider, "Towards integrated rule-driven software development for it ecosystems," in *DEST*, 2012, pp. 1–6.
[45] D. Li and Y. Yang, "Enhance value by building trustworthy software-reliant system of systems from software product lines," in *PLEASE*, 2012, pp. 13–16.
[46] D. Santamaría, F. Alarcón, A. Jiménez, A. Viguria, M. Béjar, and A. Ollero, "Model-based design, development and validation for (UAS) critical software," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 103–114, 2012.
[47] R. T. V. Braga, K. R. C. Branco, O. Trindade Jr., P. C. Masiero, L. O. Neris, and M. Becker, "The ProLiCES approach to develop product lines for safety-critical embedded systems and its application to the unmanned aerial vehicles domain," *Clei Eletronic Journal*, vol. 15, no. 8, pp. 1–13, 2012.
[48] P. Acheson, C. Dagli, and N. Kilicay-Ergin, "Model based systems engineering for system of systems using agent-based modeling," in *CSER*, 2013, pp. 11–19.
[49] S. Luna, A. Lopes, H. Yan, S. Tao, F. Zapata, and R. Pineda, "Integration, verification, validation, test, and evaluation framework for system-of-systems," *Procedia Computer Science*, vol. 20, pp. 298–305, 2013.
[50] A. Madni and M. Sievers, "System of systems integration: key considerations and challenges," *Journal Systems Engineering*, vol. 17, no. 3, pp. 330–347, 2014.
[51] R. Louali, S. Bouaziz, A. Elouardi, and M. Abouzahir, "Platform simulation based unmanned aircraft systems design," in *WCCS*, 2014, pp. 736–742.
[52] N. Kilicay-Ergin and C. Dagli, "Incentive-based negotiation model for system of systems acquisition," *Systems Engineering*, vol. 18, no. 3, pp. 310–321, 2015.
[53] ISO/IEC, "Systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models - ISO/IEC 25010:2011," British Standard, Tech. Rep., 2011.
[54] B. Boehm, "Some future trends and implications for systems and software engineering processes," *Software Engineering*, vol. 9, no. 1, pp. 1–19, 2006.
[55] 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.