The State-of-the-Art and Future Perspectives in Systems-of-Systems Software Architectures

Conference Paper · July 2013

DOI: 10.1145/2489850.2489853

5 authors, including:

Elisa Yumi Nakagawa
University of São Paulo
193 PUBLICATIONS **1,455** CITATIONS
SEE PROFILE

Milena Guessi
University of Toronto
35 PUBLICATIONS **356** CITATIONS
SEE PROFILE

Lucas Bueno Ruas de Oliveira
University of São Paulo
26 PUBLICATIONS **230** CITATIONS
SEE PROFILE

Flavio Oquendo
IRISA - Institut de Recherche en Informatique et Systèmes Aléatoires (UMR CNRS)
314 PUBLICATIONS **2,277** CITATIONS
SEE PROFILE

The State of the Art and Future Perspectives in Systems of Systems Software Architectures

Elisa Y. Nakagawa, Marcelo Gonçalves, Milena Guessi, Lucas B. R. Oliveira
University of São Paulo - USP
São Carlos, SP, Brazil
{elisa, marcelob, milena, buenolro}@icmc.usp.br,

Flavio Oquendo
IRISA-UBS - Université de Bretagne Sud
Vannes Cedex, France
flavio.oquendo@univ-ubs.fr

ABSTRACT

Currently, software systems have become increasingly large and complex, often resulted by the integration of several operationally independent systems, resulting in a new class of systems: the Systems of Systems (SoS). In another perspective, software architectures play a major role in determining system quality, since they form the backbone of any successful software-intensive system. Attention given to the software architectures of SoS is also certainly fundamental to the success of such systems. However, it is observed that there is a lack of works that present a wide and, at the same time, a detailed panorama about how SoS architectures have been treated. In this scenario, the main contribution of this paper is to present the state of the art on software architectures of SoS, mainly regarding their development, representation, evaluation, and evolution. This work also contributes with future research topics on SoS architectures that should be still investigated. Besides that, we intend this paper opens new perspectives of research in the software architecture area, intending to contribute to the success of SoS.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: [Software Architecture]

General Terms

System of systems, architectural design, architectural representation, architectural evaluation, architectural evolution

1. INTRODUCTION

Software-intensive systems have become increasing ubiquitous, larger, and complex, with considerable dissemination in various sectors and application domains. These systems have sometimes arisen as result of integration of various operationally independent systems, even developed with different technologies and for diverse platforms. An adequate integration has been more and more necessary to promote cooperation among these independent systems in order to provide more complex functions, which could not be provided by any system working separately. This class of software systems has been referred as System of Systems (SoS). There are various systems that can be classified as SoS. An example is a medical system that integrates systems to diagnosis, treatment, and management of patients [24]. Examples involving critical embedded systems can be also found, such as airport systems [28], automotive [5], avionics [19], and robotics [10]. Besides that, there are several other examples that intend to promote interoperability among several software systems in order to provide solutions to broader needs. However, in spite of their relevance, the development of SoS for software-intensive systems has not been widely investigated in the context of Software Engineering.

In another perspective, software architectures have been considered the backbone for any successful software-intensive system [35, 49] and have played a fundamental role in determining the system quality (e.g., interoperability, performance, portability, and maintainability). Decisions made at the architectural level directly enable, facilitate, or interfere with the achievement of business goals as well as functional and quality requirements. Basically, a software architecture is the structure (or a set of structures) of the system, which comprises software elements, the externally visible properties of those elements, and the relationships among them [7]. In this context, software architectures of SoS have been also noticed as an essential element to the success of such systems [12, 28, 41, 46], mainly promoting the inherent characteristics of SoS, such as the emergent behavior, geographic distribution, and evolutionary development.

Considering that software architectures for SoS is currently a new, important research area, investigation and establishment of a panorama about how such architectures have been treated is certainly interesting. The main objective of this paper is to present the state of the art on software architectures of SoS, mainly regarding construction, representation, evaluation, and evolution of such architectures. Besides that, we delineate open research issues regarding architectures of SoS. As a result, we intend this paper supports identification of other new research topics, which could also contribute to the development of SoS.

The remainder of this paper is organized as follows. Section 2 presents background involving two main topics: SoS and Software Architecture. Section 3 presents the state of the art of research that has been conducted in SoS software architectures. Section 4 presents perspectives of research in the context of SoS software architecture. Finally, Section 5 presents our conclusion and future work.

2. BACKGROUND

The term System of Systems (SoS) has been widely used in diverse domains and widespread in nature (e.g., human body, biology, and ecology) and artificial systems (e.g., computer, engineering, and society). More recently, it has been also adopted in the software domain. In general, they can be considered as a set or arrangement of independent, useful systems integrated into larger systems that deliver unique capabilities [17]. Nevertheless, the term SoS is itself very wide; Maier [41] and Firesmith [20] points that almost all systems can be understood as an SoS. Besides that, there are several works that collect and analyze definitions of SoS [20, 28, 37, 48]. There is also essentially a number of characteristics floating on different proposed definitions. Considering these works, we have noticed that the initial set of inherent characteristics of SoS, previously proposed by Maier [41], still remain referenced in most works in the context of software-intensive systems. New characteristics are variations of this set or are attached to specific application domains. The initial set of the main characteristics of SoS are [41]: (i) Operational Independence: all constituents of SoS can often deliver their functionalities when not working with other constituents; (ii) Managerial Independence: each constituents of SoS can keep its own managerial sphere; (iii) Evolutionary Development: functions and purposes of SoS can dynamically change and new constituents can be reassembled to perform them; (iv) Emergent Behaviour: SoS are capable to deliver new functionalities that are result of the constituents working together; this is the core characteristic of SoS; and (v) Geographic Distribution: constituents of an SoS are sometimes geographically distributed.

SoS started to gain their popularity mainly on military systems as a strategy to reach goals or deliver unique capabilities wherein a collaborative work of already existing and complex systems is needed [18, 41]. Currently, considering the required interconnection among software systems, as well as the growing presence of software in systems engineering, SoS has also become focus of interest in Software Engineering [9]. Evolution of computational systems points out that more and more software systems could be characterized as SoS. Nowadays, diverse application domains have presented SoS, such as office [14], telemedicine [44], and e-business [53]. It is also important to mention that SoS has been also applied in critical domains [5, 19]. For instance, Farcas *et al.* [19] present a complex software-intensive SoS for automotive and avionics. Aoyama [5] also presents an SoS for automotive embedded systems and reports that it is a new perspective to a more effective architecture than conventional functional component-and-connector model. It is observed that SoS can provide a suitable solution for a diversity of application domains where software-intensive systems are necessary and, as a consequence, SoS could certainly contribute to several sector of the society [12, 46]. Therefore, attention to the development of SoS must be intensified, including new approaches to develop and maintain them.

In another perspective, it is observed that from the first work of Kruchten on iterative software development with a focus on software architecture [33], a number of works have recognized the value of explicitly considering software architectures in the system development processes [7, 34, 52]. In the end of 90's, Bass [8] defined an architecture-centric process, focused on a set of architectural requirements in addition to functional requirements. In general, these and other processes presented a similar set of steps: requirement identification, creation/selection of the architecture, representation/communication of the architecture, analysis/evaluation of the architecture, and implementation of the system. In this scenario, other works specifically encompassing the design of software architectures were proposed. Perhaps the main ones are: Attribute-Driven Design (ADD) [6], Siemens 4 Views (S4V) [26], Rational Unified Process 4 + 1 views (RUP 4+1) [34], Business Architecture Process and Organization (BAPO) [4], and Architectural Separation of Concerns (ASC) [45]. Hofmeister et al. [25] also proposed a general model of software architecture design that synthesizes the main, common steps to build architectures: architectural analysis, architectural synthesis, and architectural evaluation.

Regarding architectural description, an adequate representation of software architectures in order to make them understandable for a wide variety of stakeholders (such as customers, product managers, project managers, and engineers) consists of an essential activity for the success of such architectures. In this context, an international standard — ISO/IEC/IEEE 42010 [27] — has supported the architectural description of software architectures and has specified minimum requirements on Architecture Description Languages (ADL). In parallel, several ADL have been proposed and widely used. In general, these languages can be characterized as formal, semi-formal, or even informal. Good, classic examples of formal languages are ACME [2], Wright [3], and Rapide [40]. Sometimes considered as a semi-formal language, UML (Unified Modelling Language) has been more and more adopted to represent software architectures, mainly in the context of industry. Even in critical domains, such as embedded systems, UML and its derivations appear [23]. Advantages can be found when using formal or semi-formal languages. Otherwise, informal representation, i.e., box-and-line drawings, can be also found representing software architectures. In short, it provides useful documentation; however, its informality limits the usefulness of the architecture description. With respect to evaluation of software architectures, several methods have been proposed and two main examples are SAAM (Scenario-Based Architecture Analysis Method) [29] e o ATAM (Architecture Trade-off Analysis Method) [31]. These methods have been updated and sometimes adapted to specific type of architectures and contexts. Moreover, evolution of software architectures has been an important topic of research, which has been widely investigated. For instance, Breivold et al. [11] present an overview, comparison, and detailed treatment of the various approaches to evolving software architectures.

It is observed that there is a considerable amount of research conducted involving software architectures, mainly regarding their design, representation, and evaluation. It is observed that this accumulated experience should be more and more reused and extended to software architectures for SoS. SoS present challenges to be built and maintained, as they present specific characteristics that must be attempted and inserted in their architectures. Following, we present the state of the art on the research of SoS software architectures.

3. THE STATE OF THE ART

In order to establish the state of the art on software architecture for software-intensive SoS, we have conducted a literature review, using the systematic review technique [32]. In short, this technique makes possible to explore, organize, and summarize contributions related to a topic of interest, providing a detailed overview by assessing the quantify of evidences existing on this topic. This review had as main objective to identify the current research involving software architectures for software-intensive SoS, i.e., how such architectures have been designed, represented, evaluated, and evolved. Furthermore, we also were interested in extracting more general information, such as how recent is this research topic and application domains that present SoS. For this, the Research Questions (RQ) of our review were: (i) RQ 1: Which are the characteristics (features and quality attributes) of SoS architectures? (ii) RQ 2: How have SoS architectures been represented? (iii) RQ 3: How have SoS architectures been evaluated? (iv) RQ 4: How SoS software architectures have been built and for which application domains? and (v) RQ 5: How SoS software architectures have been evolved?

After defining the systematic review protocol, we have conducted automatic searches on all major publication databases (ACM Digital Library, IEEE Xplore, ISI Web of Science, Science Direct, SCOPUS, and SpringerLink), resulting in 196 publications to conference and journal papers (i.e., primary studies). This search was executed in February 2013. After reading the title, abstract, and keywords of these publications and applying the selection criteria defined in the protocol, we have identified 93 publications. Following, we have read the full text of these publications and have identified 60 publications[1] that meet with the topic investigated in this work, i.e., software architectures for software-intensive SoS. This systematic review was conducted considering a high level of accuracy, as well as our previous experience in conducting reviews. We can say that results achieved in this systematic review are reliable and can represent the state of the art of architecting software-intensive SoS.

It is worth observing that most of these publications were recently published, concentrating mainly in the last seven to eight years. Figure 1 presents distribution of publications through years. From a total of 60 publications, almost 90% were published from 2004 to 2012, what can indicate an effective growth in the interest for that research area. If we consider it a trend, we can foresee for the next years a considerable number of contribution for the development of SoS.

Figure 2 shows the main application domains that have designed software architectures for their SoS. This figure presents only applications domains that had at least two publications that mention them. In addition, 70% of the total of publications has presented at least an SoS for a given domain, for instance, in a case study or an industry application. It is important to say that several publications also presented SoS in more than one application domain. Through our analysis, we have found that the predominant domain is the military area, i.e., 30% of a total of 60 publications. This

[1]The complete list of these publications can be found in: `http://www.icmc.usp.br/~elisa/2013_SESoS`

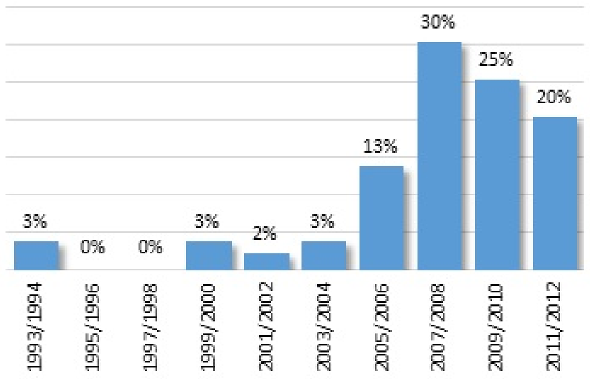

Figure 1: Publication distribution through years

can be explained by the fact that military systems are often quite complex and frequently require to be developed as an SoS. Other domains where SoS are frequent are aerospace, avionics, trading, and automotive. Hence, it is important to observe that SoS have been developed in critical domains in which quality is essential. Therefore, attention given to their architectures is also essential and could comply with the quality requirements of such systems.

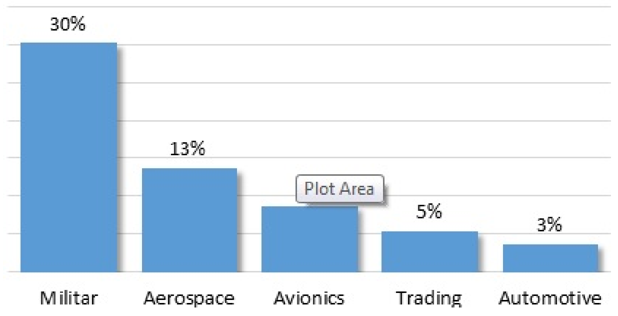

Figure 2: Application Domains

In the same perspective, most publications were concerned with the quality of SoS. In this sense, regarding quality attributes, we observed that several ones were addressed in the included publications. Figure 3 presents these attributes and their occurrences in these publications. It is interesting to observe that interoperability is by far the most addressed, i.e., it appeared in 65% of a total of 60 publications. This can be explained because SoS usually require integration of several independent systems, which must work in a synchronized way. Besides interoperability, considering their occurrence, some quality attributes that can be considered important for SoS are in this order: adaptability, security, reliability, performance, integrability, and safety. This set of attributes can provide directions of what is important, or even essential, when developing and evaluating SoS. Considering the application domains found in our systematic review, we can say that this set seems to be representative of SoS for these domains. However, this set of attributes can be different, including a different distribution, if other domains are considered in isolation, such as solely information systems.

We have been also interested in how SoS software archi-

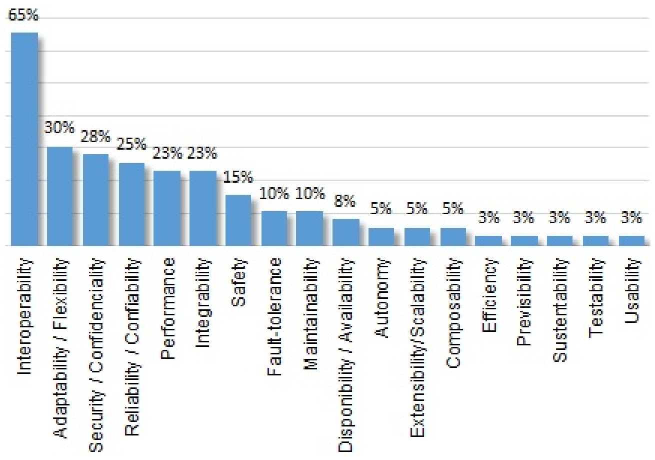

Figure 3: Quality Attributes of SoS Software Architectures

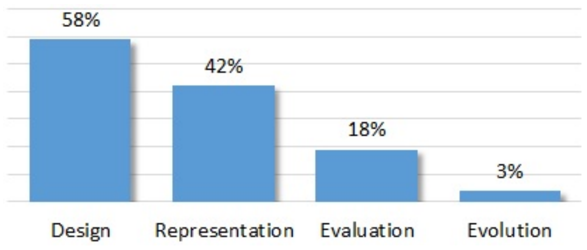

Figure 4: Research Topics on SoS Software Architecture

tectures have been built, concentrating in how they have been designed, represented, evaluated, and evolved. Figure 4 shows the distribution of 52 publications (of a total of 60), which report these topics: (i) design: it refers to approaches (including processes, methods, and guidelines) that have been used to support building of SoS software architectures; (ii) representation: it refers to the means to represent the architectural description of SoS software architectures; (iii) evaluation: it is related to means that have been adopted to evaluate SoS software architectures; and (iv) evolution: it refers to approaches that have been proposed to support evolution of software architectures of SoS. There are also publications that involve more than one topic. For instance, the work of Lewis and Feiler [38] involves three topics (design, representation, and evaluation). In particular, this work presents an ADL that supports a full model based engineering lifecycle, making possible to design, represent, and evaluate embedded software runtime architectures of software-intensive SoS.

It is interesting to observe that the majority of publications (i.e., 58%) concerned with design of SoS. Moreover, several publications were found supporting representation and evaluation of SoS software architectures, respectively with 42% and 18%. Regarding evolution of SoS software architectures, it is worth highlighting that SoS usually evolve/adapt in runtime, mainly due to two characteristics of SoS: evolutionary development and emergent behavior. However, in spite of dealing adequately with evolution of such systems, few publications were found (only 3%). Following, we discuss in more details each one of these four topics, specially detaching quantitative value as well as more relevant information about each topic. Due to lack of space, only main publications related to each topic are referenced in this work.

3.1 Design of SoS Software Architectures

Regarding design of SoS software architectures, we have found 34 publications that have proposed diverse initiatives. Specifically with regard to architectural design processes/methods, in spite of diversity of processes/methods for software architecture, such as RUP 4+1 and BAPO, few publications (i.e., only four ones [1, 16, 39, 51]) were proposed to support the complexity associated with SoS software architectures. Considering the work of Hofmeister et al. [25] that proposed common steps and basic artifacts that should be developed during the architecting activity, we can say that none of these four initiatives present the most common steps and artifacts expected in a software architecture design approach. Therefore, there is a lack of a complete, mature processes and methods that could provide a systematic way to efficiently design architectures of software-intensive SoS.

It is also interesting to observe that Service-Oriented Architecture (SOA) has been advocated as a promising architectural style for SoS. A considerable number of publications (almost 50% related to SoS software architecture design) have explored the use of SOA. For instance, in [50], the authors developed a detailed discussion about the use of SOA in SoS and affirm that a service-oriented system is a type of SoS and, therefore, given the similarities that exist between service-oriented systems and SoS, approaches and techniques that have been developed to support identification, publishing, discovery, and governance in service-oriented system can be used to support SoS. Krüger et al. [36] proposed the use of a service-oriented infrastructure, together with an agile architecture development process, for rapid SoS integration. Similarly, in [13, 19], the authors suggest the use of SOA for the integration challenges in SoS and propose the use of a hierarchical architecture pattern called "rich services" to encapsulate various capabilities and functionalities. Therefore, SOA seems to go in the direction of becoming a "standard" architectural style for SoS.

Another recurrent topic in the context of designing architectures for SoS is MDA (Model-Driven Architecture). Besides that, there are also isolated initiatives exploring previous good experiences from the other contexts and bringing them to the SoS context. In particular, some examples are: CMMI, object orientation, aspect orientation, agile methods, reference architecture, and architectural pattern; however, no more than one publication has discussed each of these topics. In spite of relevant initiatives found in our work, effort in establishing means to efficiently design architectures for software-intensive SoS is still in fact necessary.

3.2 Representation of SoS Software Architectures

An adequate representation of SoS software architectures is also quite important to the success of such systems. In this scenario, we have also investigated how these architectures have been represented, i.e., how their architectural description has been developed. From a total of 60 publications, 42% (i.e., 25 publications) have addressed the representation of SoS software architectures. In short, formal, semi-formal, and informal representation were found. In particular, existing semi-formal languages were widely used in the architectural description of SoS (in 16 of the 27 publications analysed for this topic). Specifically, the most used were: UML and SysML (Systems Modeling Language). Semi-formal representation has advantages, mainly with regard to comprehension; however, considering that the most of SoS have been built for critical domains, such as military, aerospace, and automotive, formal techniques and languages are interesting. In spite of this, there are only five initiatives that address the use of a formal representation. For instance, Gamble and Gamble [22] extend the UNITY formal specification language, which makes possible to capture programmatic, structural, and scoping properties of SoS and enable analyses of architectural properties. Conversely, even though informal languages may not be the most appropriate, informal representations are still recurrent. It was found in seven publications. Furthermore, there is also not a consensus if existing ADL are sufficient to adequately represent SoS architectures. Besides that, new ADL that could address specifically SoS and their characteristics have not been proposed.

It is also possible to identify architectural views representing SoS architectures. We have found that the most used ones are structural and behavioral views. In order to represent these views, diverse modelling techniques were adopted. For instance, the most used ones are UML sequence diagrams, UML state diagram, Message Sequence Charts (MSC). In particular, for the military domain, DoDAF (Department of Defense Architecture Framework) and its views were widely used, as this framework is especially suited to large systems with complex integration and interoperability challenges. It is also possible to say that DoDAF is almost a "standard" framework for that domain. Finally, despite these important initiatives, there is still good opportunities of research in this direction, mainly focused in establishing a clear understanding on how to improve the description of SoS software architectures.

3.3 Evaluation of SoS Software Architectures

As evaluation of software architectures is quite important to ensure that the architectural design decisions were correctly made, this activity is also important, even essential, for SoS software architectures. We have found a total of 11 publications that address evaluation of these architectures. Among them, around 50% have explored ATAM or SAAM, two well-known software architecture evaluation methods, to the SoS architectures. For instance, Kazman et al. [30] have extended ATAM to address concerns of software architectures of SoS. In another example, Gagliardi et al. [21] presented an approach based on ATAM to identify architectural risks and quality attribute inconsistencies across the constituent systems. In general, ATAM and SAAM seems to be also adequate to SoS software architectures; however, more research must be conducted in order to investigate how these methods should be adapted and used efficiently and confidently in the SoS context. The remainder of publications have presented isolated initiatives to support SoS architecture evaluation. For instance, an interesting work is [42] where the authors introduced a mathematical model to combine non-functional requirements of SoS to their architectures and present an approach to analyse the quality of software architectures, intending verification and validation of these architectures.

In general, it can be observed that the found publications have proposed reuse of existing approaches, as well as their adaptations to the SoS context. However, there is not a consensus about what exactly should be considered when evaluating SoS architectures. Hence, evaluation of architectures of SoS still remains as an open issue to be further investigated.

3.4 Evolution of SoS Software Architectures

As stated before, one of the main characteristics of SoS is evolutionary development. Their architectures must be prepared to support dynamic evolution, making it possible to incorporate new functionalities or remove existing ones. However, we have found that, in spite of required attention to this dynamism, there is only few approaches (i.e., only two publications) that address evolution of SoS software architectures. In particular, the work of Chen and Han [15] can be considered as the most aligned with SoS evolution. These authors presented an initial proposal of an environment that intends to manage all systems involved in the SoS evolution. In another example, Selberg and Austin [47] presented key characteristics, for instance, standard interfaces and interface layers, that could become an SoS architecture as an evolutionary one. There are still only initial proposals and they should be evolved/refined in order to have a real impact on the evolution of SoS software architectures.

It is also worth observing that there is no publication that reports any observation or experience of how SoS and their architectures have evolved. We can say that either evolution has occurred without adequate, systematic control or the community has not perceived the importance of understanding and reporting SoS evolution. Therefore, evolution of SoS architecture will require considerable attention and research efforts yet.

4. PERSPECTIVES OF RESEARCH IN SOS SOFTWARE ARCHITECTURES

Considering the state of the art on software architectures for software-intensive SoS, it can be observed that it is a relatively new field of research and, therefore, considering also the importance of this field, effort to develop this field is quite necessary. There is still a considerable number of open issues that must be investigated and challenges that must be overcome. In this perspective, we have identified what could be considered potential, promising research lines in the context of SoS software architectures:

SoS architectural design approaches: SoS designers are challenged to architect systems more and more complex, which present a set of complex characteristics. It is worth highlighting that when designing SoS architectures, a new scenario must be considered. During the design of SoS, their constituents are usually unknown, as emergent behavior can bring the necessity of introducing new constituents. In other words, these constituents are unknown during the conception of the initial architecture. Besides that, these constituents are as "black-box" with interfaces that should be well-defined and are sometimes legacy systems that were developed previously for a particular context. These constituents are also geographically, managerially, and operationally distributed. In this perspective, an existing open issue is how to facilitate the construction of SoS with these diverse characteristics. For this, it is required processes containing integrated methods, techniques, and guidelines. These processes should include basic steps presented in most of architecture design processes, namely, architectural synthesis, architectural analysis, and architectural evaluation. These processes must be also widely experimented and matured and, as a consequence, widely adopted by SoS designers. Moreover, software tools and environments that could support proposed processes, methods, and techniques are also interesting, even necessary, to improve productivity during architecting of SoS.

SoS software architecture representation: Regarding representation of SoS architectures, there are also diverse topics that must be investigated yet. First of all, it is important to investigate if existing ADL (formal and semi-formal) are sufficient to represent SoS architectures. If insufficiency is observed, it is necessary to propose new ADL or to extend existing ones to adequately support representation, evaluation, and evolution of dynamic architectures of SoS. Currently, semi-formal ADL, in particular, UML and its derivations, have been a trend; however, considering critical characteristics of SoS (e.g., emergent behaviour, evolutionary development, and geographic distribution), formal techniques and languages must be widely introduced in order to adequately represent SoS architectures. It is also necessary to define which level of formalism is necessary and also in which situation this formalism should be considered. Besides that, these ADL could be generic (i.e., destined to any type of SoS) or specific (i.e., for a given type or application domains). At the end, it is a need to get empirical evidences about viability and advantages of semi-formal and formal ones with regard to these ADL.

With respect to architectural viewpoints and views, it is interesting to conduct an investigation about which ones are relevant or sufficient for SoS. Moreover, it is also interesting to establish with ADL or its adaptation could better cover such viewpoints and views. As a consequence, a "standard" for the architectural descriptions could be established, mainly aiming at the understanding by organizations and partners that will develop or adapt their systems to be part of an SoS.

Evaluation of SoS software architectures: As stated before, evaluation is essential during building of any software architecture. In particular, SoS architectures should be evaluated not only during their building but also during evolution that occurs sometimes under execution. During building (that commonly occurs after architectural analysis and synthesis), these architectures could be evaluated using existing approaches, such as ATAM and SAAM, as well as their adaptations to the SoS context. Otherwise, the inherent dynamism of these architectures requires new approaches that could ensure that changes in the architectures do not degrade them, mainly regarding to quality attributes, such as interoperability, adaptability, and performance. In this context, experience with existing approaches for evaluation of dynamic architectures should be investigated and, if possible, reused in the SoS context. In this research perspective, processes, methods, and techniques for evaluation of dynamic software architectures of SoS need to be proposed and also widely adopted. Besides that, software tools that automate tasks related to evaluation will be also certainly welcome.

Considering the set of quality attributes found in our work, quality models for SoS software architectures could be established. These models could contain what are considered essential quality attributes, such as interoperability and reliability, and the relationship among them. It is also interesting to establish quality models for specific types of SoS (e.g., service-oriented SoS and network centred SoS) or domains (e.g., embedded systems and information systems), since this diversity certainly influences the importance of each attribute. These quality models could guide the evaluation and even the design and evolution of SoS architectures, resulting in quality-based processes, methods, and techniques.

Evolution of SoS software architectures: The vast majority of SoS evolve by nature. Simply ignoring the need for controlling their evolution is not certainly the most adequate way. Hence, it is firstly important to distinguish a stand-alone system's evolution from SoS evolution and to treat SoS evolution as part of the whole development process supported by proper methodologies and techniques. Moreover, it is important to understanding why, how, when, and where this evolution occur and which characteristics and decisions (e.g., external interfaces of the SoS constituents) must be taken when proposing SoS architectures, in order to avoid future architecture degradation. If it is intended to keep quality in SoS along their evolution, new processes, methods, and techniques that systematize SoS evolution must be investigated, proposed, and widely adopted. Additionally, it is essential to have integrated environments and tools, mechanisms, and technologies that could transparently evolve SoS and, as a consequence, to manage their dynamic architectures and to deal with complicated SOS evolution challenges.

Reference architectures for SoS: SoS usually involves diverse software systems, different technologies, and even several organizations that apply different approaches to develop these systems. In this context, the importance of reference architectures emerges. Reference architectures refer to a special type of software architecture that aggregates knowledge of architectures of a set of systems of a given domain [43]. Considering advantages of such architectures for the development, standardization, and evolution of software systems, establishment of reference architectures for diverse domains in which SoS have been developed is certainly interesting. For instance, a reference architecture for automotive domain could be developed. This reference architecture could propose, for instance, the standard interfaces used for communication among all systems of an automotive system, in the same perspective of AUTOSAR. Despite AUTOSAR can be considered as a reference architecture for SoS [19]; it has been proposed without completely considering characteristics and challenges to develop and evolve SoS. Probably the current automotive fleet, which had its systems developed based on AUTOSAR, does not have systems with dynamic architectures, i.e., new services can be incorporated and old one can be discarded while systems are running (in other words, after cars were acquired). AUTOSAR and other important reference architectures could be analysed from this new perspective. Furthermore, in order to adequately deal with reference architectures for SoS, it is also necessary to propose means to design, represent, evaluate, and evolve such architectures. Besides that, as SOA has been in some way widely investigated to be basis of SoS, it could be also considered to compose reference architectures for this class of systems.

Development of SoS for Diverse Domains: Currently, according with results of our systematic review, we have found few application domains that have developed SoS with concern in their software architectures. However, there are many other domains that could be beneficed with research on software architectures for SoS. In other words, other application domains should also explore concepts and principles of SoS in order to organize and make available their systems. Some examples of domains that could have software systems developed under SoS characteristics are enterprise information systems making possible to integrate and coordinate enterprise business processes, distributed computer games running in diverse platforms, such as tablets and computers, and software engineering environments integrating several distributed tools and promoting a global software development.

In this work, we intend to trigger a first discussion about the possible lines of research in software architecture for software-intensive SoS, which is already a considerable amount of research that could be conducted in this area. However, we believe that there are also other research lines as important as those presented here.

5. CONCLUSION AND FUTURE WORK

SoS have recently become increasingly large-scale, complex software systems. Their software architectures have also become the heart of the development of such systems. In this scenario, the main contribution of this paper was to present the current state of the art on how software architectures of SoS have been designed, represented, evaluated, and evolved. For this, we adopted the systematic review technique. As main result, we have found several initiatives to deal with SoS software architectures; however, in general, they are not enough mature, adequately adapted, and widely adopted for such architectures. We have also found important trend in this area, such as the adoption of SOA as an architectural style for SoS and reuse of previous approaches that have worked in other contexts and adaptation to SoS architectures. Otherwise, we have observed that important topics have not been investigated as they should be. It is important to detach that evolution of SoS architectures is the main one, considering the necessity of managing dynamic architectures that meet with inherent characteristics of SoS, i.e., the emergent behavior and evolutionary development. Therefore, in general, the research area is a relatively new one with its publications concentrated in the last years. We have also observed that the field of SoS software architectures presents several lines of research that deserve to be further investigated. Among these lines, in this paper, we have detached some ones that we believe to be more urgent. For instance, the proposal of complete, integrated processes that make possible to design, represent, evaluate, and evolve SoS software architectures seem to be interesting for this first moment. In view of this fact, following the trend, we will have certainly good contributions in near future. For future work, we intend to continually update this state of the art in order to contribute to the area of SoS and, considering the need and relevance of these systems, we intend this state of the art contributes to the Software Engineering community that need to adequately develop software-intensive SoS.

Acknowledgments: This work is supported by the Brazilian funding agencies FAPESP, Capes, and CNPq.

6. REFERENCES

[1] P. Acheson. Methodology for object-oriented system architecture development. In *SysCon'2010*, pages 643–646, San Diego, USA, 2010.

[2] ACME. The Acme Project. [*On-line*], *World Wide Web*, 2013. Available in: `http://www.cs.cmu.edu/~acme/` (Access in 05/28/2013).

[3] R. Allen. *A Formal Approach to Software Architecture.* PhD thesis, Carnegie Mellon, School of Computer Science, Jan. 1997.

[4] P. America, E. Rommes, and H. Obbink. Multi-view variation modeling for scenario analysis. *Software Product-Family Engineering*, pages 44–65, 2004.

[5] M. Aoyama and H. Tanabe. A design methodology for real-time distributed software architecture based on the behavioral properties and its application to advanced automotive software. In *APSEC'2011*, pages 211–218, Ho Chi Minh, Vietnam, 2011.

[6] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice.* 2003.

[7] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice.* Addison-Wesley, 2012.

[8] L. Bass and R. Kazman. Architecture-based development. Technical Report CMU/SEI-99-TR-007, SEI, Pittsburgh, USA, 1999.

[9] B. Boehm and J. Lane. 21st century processes for acquiring 21st century software-intensive systems of systems. *The Journal of Defense Software Engineering*, 19(5):4–9, 2006.

[10] R. Bowen and F. Sahin. A net-centric xml based system of systems architecture for human tracking. In *SoSE'2010*, pages 1–6, Loughborough, UK, 2010.

[11] H. P. Breivold, I. Crnkovic, and M. Larsson. A systematic review of software architecture evolution research. *Information and Software Technology*, 54(1):16 – 40, 2012.

[12] J. Brondum and Z. Liming. Towards an architectural viewpoint for systems of software intensive systems. In *SHARK '2010*, pages 60–63, Cape Town, South Africa, 2010.

[13] P. Bull, A. Grigg, L. Guan, and I. Phillips. A quality of service framework for adaptive and dependable large scale system-of-systems. In *SoSE'2010*, pages 1–6, Loughborough, UK, 2010.

[14] R. Carbon, G. Johann, D. Muthig, and M. Naab. A method for collaborative development of systems of systems in the office domain. In *EDOC'2008*, pages 339–345, Munich, Germany, 2008.

[15] P. Chen and J. Han. Facilitating system-of-systems evolution with architecture support. In *IWPSE'2001*, pages 130–133, Vienna, Austria, 2001.

[16] A. Chigani and O. Balci. The process of architecting for software/system engineering. *International Journal of System of Systems Engineering*, 3(1):1–23, 2012.

[17] DoD. *System Engineering Guide for Systems of Systems.* Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering, 2008. Version 1.0.

[18] DoD - Department of Defense. The DoDAF architecture framework version 2.02. [*On-line*], *World Wide Web*, 2010. Available in `http://cio-nii.defense.gov/sites/dodaf20/`

(Access in 05/28/2013).

[19] 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*, 98(4):562–583, 2010.

[20] D. Firesmith. Profiling systems using the defining characteristics of systems of systems (SoS). Technical report, Software Engineering Institute (SEI), 2010.

[21] M. Gagliardi, W. G. Wood, J. Klein, and J. Morley. A uniform approach for system of systems architecture evaluation. *CrossTalk*, 22(3-4):12–15, 2009.

[22] M. Gamble and F. Gamble. Reasoning about hybrid system of systems designs composition-based software systems. In *ICCBSS'2008*, pages 154–163, Madrid, Spain, 2008.

[23] M. Guessi, E. Y. Nakagawa, F. Oquendo, and J. Maldonado. Architectural description of embedded systems: a systematic review. In *ISARCS'2012*, pages 31–40, Bertinolo, Italy, 2012.

[24] Y. Hata, Y. Kamozaki, T. Sawayama, K. Taniguchi, and H. Nakajima. A heart pulse monitoring system by air pressure and ultrasonic sensor systems. In *SoSE'2007*, pages 1–5, 2007.

[25] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America. A general model of software architecture design derived from five industrial approaches. *Journal of Systems and Software*, 80(1):106 – 126, 2007.

[26] C. Hofmeister, R. Nord, and D. Soni. *Applied software architecture*. Addison-Wesley, 2000.

[27] ISO. ISO/IEC 42010 - Systems and software engineering – Architecture description, 2011.

[28] M. Jamshidi. *System of Systems Engineering - Innovations for the 21st Century*. John Wiley & Sons, New York, 2008.

[29] R. Kazman, L. Bass, G. Abowd, and M. Webb. SAAM: A method for analyzing the properties of software architectures. In *ICSE'1994*, pages 81–90, Sorrento, Italy, May 1994.

[30] R. Kazman, M. Gagliardi, and W. Wood. Scaling up software architecture analysis. *Journal of Systems and Software*, 85(7):1511–1519, 2012.

[31] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere. The architecture tradeoff analysis method. In *ICECCS'1998*, pages 68–78, Monterey, USA, Aug. 1998.

[32] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele Univ. and Durham Univ., 2007.

[33] P. Kruchten. An iterative software development process centered on architecture. In *Proc. 4eme Congres de Genie Logiciel*, pages 369–378, Toulouse, France, 1991.

[34] P. Kruchten. *The Rational Unified Process: An Introduction*. The Addison-Wesley Object Technology Series. Addison-Wesley, 3 edition, 2003.

[35] P. Kruchten, H. Obbink, and J. Stafford. The past, present, and future for software architecture. *IEEE Software*, 23(2):22–30, 2006.

[36] I. Kruger, M. Meisinger, M. Menarini, and S. Pasco. Rapid systems of systems integration - combining an architecture-centric approach with enterprise service bus infrastructure. In *IRI'2006*, pages 51–56, Waikoloa, USA, 2006.

[37] J. Lane and R. Valerdi. Synthesizing SoS concepts for use in cost estimation. In *SMC'2005*, volume 1, pages 993–998, Hawaii, USA, 2005.

[38] B. Lewis and P. Feiler. Multi-dimensional model based engineering using AADL. In *RSP'2008*, pages xv–xviii, Monterey, USA, 2008.

[39] S. X. Liang, J. F. Puett, and Luqi. Quantifiable software architecture for dependable systems of systems. *LNCS*, 3069:241–265, 2004.

[40] D. C. Luckhama, J. J. Kenney, L. M. Augustin, J. Vera, D. Bryan, and W. Mann. Specification and analysis of system architecture using rapide. *IEEE Transactions on Software Engineering*, 21(4):336–355, 1995.

[41] M. W. Maier. Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284, 1999.

[42] J. Michael, R. Riehle, and M.-T. Shing. The verification and validation of software architecture for systems of systems. In *SoSE'2009*, pages 1–6, Loughborough, UK, 2009.

[43] E. Y. Nakagawa, F. Oquendo, and M. Becker. RAModel: A reference model of reference architectures. In *WICSA/ECSA'2012*, pages 297–301, Helsinki, Finland, 2012.

[44] V. Petcu and A. Petrescu. Systems of systems applications for telemedicine. In *RoEduNet'2010*, pages 208 –211, Sibiu, Romania, 2010.

[45] A. Ran. ARES conceptual framework for software architecture. *Nokia Research Center*, 2000.

[46] R. Schaefer. Systems of systems and coordinated atomic actions. *SIGSOFT Softw. Eng. Notes*, 30(1):6–11, 2005.

[47] S. Selberg and M. A. Austin. Toward an evolutionary system of systems architecture. Technical report, Institute for Systems Research, University of Maryland, USA, 2012.

[48] A. Sharawi, S. Sala-Diakanda, A. Dalton, S. Quijada, N. Yousef, L. Rabelo, and J. Sepulveda. A distributed simulation approach for modeling and analyzing systems of systems. In *WSC'2006*, pages 1028–1035, Monterey, USA, 2006.

[49] M. Shaw and P. Clements. The golden age of software architecture. *IEEE Software*, 23(2):31–39, Mar/Apr 2006.

[50] S. Simanta, E. Morris, G. A. Lewi, and D. B. Smith. Engineering lessons for systems of systems learned from service-oriented systems. In *SysCom'2010*, pages 634–639, San Diego, USA, 2010.

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

[52] Z. You-Sheng and H. Yu-Yun. Architecture-based software process model. *ACM SIGSOFT Software Engineering Notes*, 28(2):1–5, Mar. 2003.

[53] L. Zhu, M. Staples, and R. Jeffery. Scaling up software architecture evaluation processes. In *ICSP'2008*, pages 112—122, Leipzig, Germany, 2008.