Submission date: 17/12/2018
Resubmission date: 20/04/2019
Camera ready submission date: 19/08/2019

1st round notification: 26/02/2019
2nd round notification: 13/06/2019
Available online: 19/08/2019
Publication date: 22/10/2019

Section: thematic issue, extended version
Social, Human and Economic Aspects of Software

S.O.B. (Save Our Budget) - A Simulation-Based Method for Prediction of Acquisition Costs of Constituents of a System-of-Systems

Valdemar V. Graciano Neto[1], Flávio E. A. Horita[2], Rodrigo Pereira dos Santos[3], Davi Viana[4], Mohamad Kassab[5], Wallace Manzano[6], Elisa Yumi Nakagawa[6]

[1]Universidade Federal de Goiás, Goiânia – GO – Brazil

[2]Universidade Federal do ABC, Santo André – SP – Brazil

[3]Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro – RJ – Brazil

[4]Universidade Federal do Maranhão, São Luis – MA – Brazil

[5]Pennsylvania State University, Malvern – PA – United States

[6]Universidade de São Paulo, São Carlos – SP – Brazil

valdemarneto@inf.ufg.br, flavio.horita@ufabc.edu.br, rps@uniriotec.br

davi.viana@lsdi.ufma.br, muk36@psu.edu, wallace.manzano@usp.br, elisa@icmc.usp.br

Abstract. *Software economics, acquisition, and pricing are important concerns for Systems-of-Systems (SoS). SoS are alliances of independent software-intensive systems combined to offer holistic functionalities as a result of the constituents interoperability. SoS engineering involves separately acquiring constituents and combining them to form the SoS. Despite the existence of cost prediction techniques, predicting SoS acquisition costs at design-time should also include the analysis of different suppliers of constituents, their respective prices and quality. However, known methods cover only two out of these three parameters. The main contribution of this article is to present the S.O.B. (Save Our Budget) method, a novel simulation-based method to predict, at design-time, the acquisition cost of constituents, while still considering quality attributes and different suppliers. Results of a case study in the Smart Building domain revealed that S.O.B. method supports a precise prediction of acquisition cost of constituents to build a SoS for that domain. Furthermore, it also contributes to estimate the cost based on a pre-established quality attribute (functional suitability), as well as to support the selection of coalition that exhibits better results through the analysis of cost-benefit ratio.*

Keywords. *Acquisition, Cost, Prediction, Architecture, System-of-Systems, Evaluation, Quality Attribute*

Cite as: Graciano Neto, V. V., Horita, F. E. A., Santos, R. P., Viana, D., Kassab, M., Manzano, W., & Nakagawa, E. Y. (2019). S.O.B. (Save Our Budget) - A Simulation-Based Method for Prediction of Acquisition Costs of Constituents of a System-of-Systems. iSys: Revista Brasileira de Sistemas de Informação (Brazilian Journal of Information Systems), 12(4), 06-35.

1. Introduction

Software-intensive Information Systems (IS) are the cornerstone of modern companies, which often interoperate their systems with external systems and/or technologies, such as drones and security cameras, to create innovative business models. With the emergence of smart-* (e.g., smart cities and smart farms), managers often rely on systems' acquisition, software, and hardware, such as smart sensors, alarms, and smart control systems. On one hand, companies can compete for selling such systems by establishing competitive prices; on the other hand, a buying manager can build a positive decision to buy a system if the specification requirements are matched with the lowest price.

However, pricing and acquisition processes for these emerging systems have faced some additional challenges, including: (i) the acquisition of multiple systems (e.g., flood monitoring systems and smart traffic systems); (ii) different available suppliers[1]; and (iii) guarantee of their compatibility, interoperability, overall performance, and a trade-off between cost and functionalities provided. Besides, the selection of the systems that are trully required to form the SoS is also a concern.

These concerns are important because the systems have been put together to form what is known as Systems-of-Systems (SoS[2]). SoS comprise many independent software-intensive systems, known as constituents, which are combined to provide complex functionalities that could not be offered individually by their constituents. Since SoS depend on the compatibility among their constituents to achieve a cohesive mission, the design of a SoS should involve a careful selection of the participating constituents that exhibit the desired capabilities [Burton et al. 2014] and best results to contribute to the accomplishment of pre-established missions [Silva et al. 2015]. However, several candidate constituents may offer similar functionalities; hence, it is important to consider important factors such as the cost, which is the main criterion to drive decisions on acquisition of systems and predict how they will influence in the SoS holistic performance.

Acquisition of systems to compose a larger set of interoperable systems is not a new trend. It has occurred since the 1970s in the USA, especially in the military domain [Acker 1983]. Satellites, airplanes, missiles, and systems have been purchased to interoperate for a long time during the last decades. However, these constituents are often individually acquired without: (i) a thorough investigation on the value delivered when integrated within a larger system; (ii) a guarantee of functional compatibility; (iii) a thorough investigation on the architectural configurations required to optimize the overall results; (iv) a determination of the number of constituents effectively needed to solve a problem; and (v) the quality yielded by different architectural arrangements that can be obtained by the varying number, types, and suppliers of constituents. Evaluating costs and benefits in the SoS context can be a complex task, since during its execution, a SoS can assume several distinct architectural configurations, which present different results that can influence the number of constituents required to be acquired, and the arrangement

[1]Herein, we adopt the term 'supplier' to denote the names of the different industrial manufacturers that build and sell constituent systems, e.g., Raspberry Pi; or to denote the name of product itself, e.g. Arduino.

[2]For sake of simplicity, along this text, this acronym will be interchangeably used to express both singular and plural forms.

that should be kept during SoS operation. Decisions made in the software development processes, especially in software architecture, have economic implications on the cost perspective. Therefore, it is important to investigate this economic aspect.

This article presents an extension and consolidation of the results previously obtained. The presented method, S.O.B. (*Save our Budget*), is an extension of a previous method called ASAS [Graciano Neto et al. 2018a],which is a simulation-driven and model-based method for analysis of SoS architectures. ASAS allowed to draw conclusions about the better architectural configurations using specific parameters, such as the success to deliver the expected behaviors. ASAS was comprised of the following steps: (i) SoS Architectural specification in SosADL, (ii) Model transformation execution, (iii) Simulation execution, and (iv) Coalitions analysis. S.O.B. method, presented in [Graciano Neto et al. 2018a],enriches ASAS by adding a new step (cost estimation) to the workflow using the results obtained as outcome from the architectural analysis step. Besides that, in the previous studies [Graciano Neto et al. 2018c, Graciano Neto et al. 2018a], we reported results using a Urban Flood Monitoring SoS. Herein, we recall the S.O.B. method and conducted the study in different application domain. A novel architectural specification for a Smart Building SoS was developed from scratch and analyzed for cost prediction. Results show the S.O.B. method allowed us to perform a successful trade-off analysis and reach a balance between cost and quality offered by that SoS. In the analyzed instance, the S.O.B. method provided support for a decision maker to choose between (i) a cheaper architectural arrangement (6K dollars) with a reasonable performance (70% efficiency to deliver its functionalities), (ii) a more expensive arrangement (12K dollars) with performance close to 100%, and to decide that the most expensive arrangement is not worth since it costs 22K dollars and performance of 92% (lower than the second arrangement). We conclude S.O.B. method subsidizes users to decide the best SoS architectural arrangement by addressing a precise trade-off analysis between cost and quality effectively delivered.

The article is structured as follows. Section 2 presents the foundations to understand the S.O.B. method. Section 3 details our method, while Section 4 shows results of an evaluation of S.O.B. method and discusses our results. Finally, Section 5 draws conclusions and indicates future work.

2. Background and Related Work

SoS comprise a set of operationally and managerially independent systems combined to offer larger functionalities that could not be individually delivered by any of them [Maier 1998]. Such complex functionalities are materialized as intended emergent behaviors, which can be intentionally engineered to accomplish a pre-defined set of missions [Rodriguez and Nakagawa 2017]. Individual missions are realized by constituent systems themselves whereas global missions of an SoS are accomplished through emergent behaviors [Silva et al. 2015]. SoS fulfill global missions by: (i) performing assigned activities (individual missions) through constituents capabilities; and (ii) interacting constituent systems leading to emergent behaviors.

The software architecture of a single software system comprises the fundamental structure of that system, contaning software elements, relations among them, and

the rationale, properties, and principles governing their design and evolution [ISO 2011, Bass et al. 2012]. In turn, a SoS software architecture involves its fundamental structure, which includes its constituents and connections among them, their properties, as well as those of the surrounding environment [Nielsen et al. 2015]. SoS software architectures are highly dynamic, i.e., they continuously change at runtime in response to addition, substitution, and deletion of constituents [Cavalcante et al. 2015]. In SoS software architectures, an *architectural configuration* is the current state and organization of an arrangement of interoperable software-intensive systems at a given point of time, also known as *coalition*. During the SoS operation, its software architecture can assume many architectural configurations due to its *dynamic architecture* property. Each architectural configuration yields specific values about performance, reliability, and effectiveness. Such values can be collected through simulations, which enable an architect to anticipate, at design-time, the structure and behavior of a SoS before being deployed [Graciano Neto et al. 2018c]. Once a better configuration is achieved, i.e., those systems that exhibit better results with the lowest cost (a lower number of constituents) are found, a self-healing mechanism can be triggered to maintain that coalition along the rest of the SoS operation, unless an emerging need of changing such a structure occurs. Therefore, coalitions can be predicted at design-time through simulations, and deployed to work later. Hence, the cost of system acquisition can be calculated in function of the predicted set of necessary (and enough) constituents, besides a margin of replacement (such as 10% of extra constituents) in case of defects or need of substitution.

One important concern for a SoS is its functional suitability. This prominent quality attribute is related to the degree to which a SoS provides functions (behaviors) that meet stated and implied needs when used under specified conditions [ISO/IEC 2011]. This is an important quality attribute when a government or an individual intends to acquire constituents to be part of a SoS, since the individual results provided by a constituent can impact the entire SoS, and the entire SoS can exhibit different functional suitability depending on the different coalitions and different suppliers involved.

Cost estimation prediction have been largely discussed in software engineering literature [Akintoye and Fitzgerald 2000, Boehm et al. 2000, Moløkken-Østvold et al. 2004, Yang et al. 2008, Sharma et al. 2012]. However, the majority of the approaches, such as SLIM, PRICE-S, SEER, and COCOMO, relies on estimation of effort to develop new software [Boehm et al. 2000]. Conversely, in regards to SoS engineering, this process is often converted in a Cost Prediction process for Software-Intensive System Acquisition, since we draw a mission composed of many goals that should meet a set of required capabilities. The software-intensive constituent systems should then be acquired (together with the hardware) based on their required capabilities to be capable of achieving the set of established missions, as highlighted by the US Department of Defense [Olagbemiro et al. 2009].

2.1. Related Work

Adopting simulations to support cost estimation is not a novel trend. Several studies have been conducted over the past decades, although the most of them were not conducted in the context of SoS [Yang 2005, Asiedu and Besant 2000]. A search using the

string `"simulation" AND "cost" AND "systems of systems"`[3] returned only eight studies in IEEE Xplore[4], seven studies in ACM Digital Library[5] and only 75 in Google Scholar[6] on April 6th, 2019[7]. Acquiring constituents to form such SoS depends on a manifold analysis: (i) selection of constituents that offer the required set of capabilities necessary to fulll the pre-established missions, (ii) assessment of the coalitions that offer better results, (iii) quality attributes such as performance, and (iv) the available budget. Hence, constituents acquisition inherently involves a cost-benet trade-off analysis, i.e., a balance between the cost associated to a product and quality offered by it. Table 2.1 summarizes the comparison among related works according to the aforementioned parameters.

Study	***Selection of Constituents based on capabilities***	***Assessment of multiple coalitions***	***Analysis of quality attributes***	***Prediction of total cost***
Takakuwa 1997	✓	✗	✗	✓
Capdem 2005	✓	✗	✗	✓
Lowe and Chen 2008 (Speculatively)	✓	✓	✓	✗
TLCM 2010	✓	✗	✗	✓
Burton et al. (2012, 2014)	✓	✗	✗	✓
Ricci et al. 2013	✗	✓	✓	✗
Axelssson 2018	✗	✗	✓	✗

Table 1. Comparison between related works.

Takakuwa, for instance, conducted a simulation-based study for an accurate determination of cost of components for the operation of a exible manufacturing system (FMS), i.e., a set of manufacturing systems that control both material and information ows for production of versatile items [Takakuwa 1997]. The author relies on optimization functions to predict the total manufacturing cost as the sum of the cost of materials, labor, and applied overhead. They consider the cost accounting from the perspective of the material and labor costs, not acquisition costs, not necessarily considering the software involved or the functional suitability.

Lowe and Chen (from Boeing) discuss and emphasize the importance of applying a capability-based acquisition approach for the development of multiple alternative SoS

[3]By using *cost prediction* or *cost estimation* rather than only ”cost”, no one study was retrieved by ACM and IEEE. Hence, we prefered to use a more broad term: *cost*. Besides, ’cost’ is dealt with as a *computation cost* or *simulation cost* or *operational cost* in some studies, not monetary cost as we address herein.

[4]`https://ieeexplore.ieee.org/Xplore/home.jsp`

[5]`https://dl.acm.org/dl.cfm`

[6]`https://scholar.google.com`

[7]Since the focus of this article is not to perform a mapping study, we conducted an exploratory study using those academic basis to bring a panorama of the state of the art by discussing the studies that are close enough to our method.

architectures to link (i.e., network) diverse interoperable systems to optimize overarching capability effectiveness while minimizing development costs [Lowe and Chen 2008]. They consider simulation, alternative coalitions, quality attributes (such as effectiveness), but no evidence is provided of the approach and how they conduct it.

Ricci et al. studied eight different SoS coalitions, evaluating and comparing them in regards to four value sustainment strategies [Ricci et al. 2013]: (1) self-recovery, the SoS is not changed (i.e., relating to survivability/robustness); (2) changes in the design of the SoS are allowed (i.e., relating to changeability); (3) changes in the architecture of the SoS are allowed (i.e., relating to evolvability) once, or (4) three times in the eight years. Their results provided a quantitative approach to gain insights into trade-offs in how SoS architects can create value-sustainable SoS for the long run. Then, they analyzed some quality attributes in multiple coalitions; however, neither a total cost estimation is not provided, nor a selection of capabilities.

Axelsson recently published a work in which he reinforces that cost-benet analysis for SoS is critical and decisions involve multiple factors [Axelsson 2018]. Besides, the author claims that the challenges of SoS cost-benet analysis are in particular a consequence of the managerial independence of the constituents. Although cost-benet analysis is discussed, the author uses simulation to investigate the relation between energy and transportation efciencies in a truck highway SoS. However, cost prediction is not provided neither an assessment of multiple coalitions or capabilities.

TLCM (Through Life Capability Management) [Urwin et al. 2010] and CapDEM [Robbins et al. 2005] are examples of approaches that rely on capability-based planning for predicting acquisition cost. However, those processes do not address an anticipation of the results exhibited by those coalitions measured in terms of quality attributes.

SoS constituents acquisition processes are often based on capability-based planning approaches, i.e., an optimization procedure that searches for a good solution that balances the set of desired capabilities and potential coalitions [Burton et al. 2014]. Burton et al. (2012) adopt a Model-Driven Engineering (MDE) approach, which includes domain-specific modeling languages to automatically generate potential solutions to the acquisition problem [Burton et al. 2012]. They progressed towards visualization techniques for the proposed solutions, and trade-off analysis for acquisition [Burton et al. 2014]. However, there is no focus on the results yielded by those potential solutions, specially with regard to quality attributes such as functional suitability that was considered by S.O.B. method in the case study presented in this article.

A recent work invested on simulations for predicting attributes of a SoS software architecture at design-time [Graciano Neto et al. 2018c]. In this approach, the authors specify a SoS software architecture using SosADL models [Oquendo 2016] and automatically generate simulation models documented in DEVS [Zeigler et al. 2012]. After the assessment of multiple coalitions, the best configuration is elected. The method proposed by Graciano Neto et al. currently supports the assessment of the functional suitability of a SoS, but it does not involve cost prediction. The next section details how such approach has been exploited for the prediction of SoS acquisition cost.

3. S.O.B. Method: A Simulation-Based Method to Support Constituents Acquisition for the Systems-of-Systems Engineering

The S.O.B. method is concerned to the prediction of costs for the processes of acquisition of software-intensive constituent systems, i.e., systems intended to be part of a SoS that include hardware but have software as a dominant part as in their structure, as in their development and/or integration process [ISO 2011]. This class of systems include several complex systems, ranging from IS to SoS. S.O.B. does not consider integration costs.

S.O.B. Method was built on top of ASAS method [Graciano Neto et al. 2018c], a simulation-driven model-based approach. ASAS supports SoS and software architects to evaluate multiple coalitions and analyze which one exhibits better results considering a set of attributes previously established, such as the percentage of achievement of missions and data transmission. Originally, ASAS comprised only four primary steps: (i) SoS architectural specification in SosADL, (ii) Model transformation execution, (iii) Simulation execution, and (iv) Coalitions analysis. Then, we enriched ASAS by adding a fifth step to systematize the estimation of acquisition cost considering the trade-off analysis obtained as outcome from coalitions analysis step.

Figure 1 depicts S.O.B. method that aims to support the selection of better architectural configurations. For determining the cost of system acquisition, the method starts with a list of constituent systems that goes through the following steps:

Step 1. Specification of a SoS architecture using SosADL.[8] Firstly, SoS architecture models are specied in SosADL. To conduct this activity, it is necessary to identify the constituent systems that are intended to be part of the SoS, and how to interconnect and orchestrate them to design the intended holistic behaviors to emerge as a result of the constituents interoperability. For instance, if one intends to specify a SoS for environmental monitoring, the candidate systems are a satellite, multiple data collection platforms (DCP) with sensors for humidity, rain, temperature and others, and a center for command and control (C2) [Neto et al. 2018]. A pre-established mission (monitoring the environmental conditions of Amazon) drives the combination of the constituents to reach the goal. DCP are placed in strategic positions and when a satellite ies over them, the data are uploaded to it, and later downloaded to C2 when the satellite ies over it. Those models are then specied in SosADL, documenting the individual structure and behavior of each system and how they exchange data;

Step 2. Model transformation execution. SosADL models are used as input of a model transformation that automatically generates simulation models specied in DEVS (a discrete event simulation formalism)[9].

Classic DEVS models are based on atomic and coupled models. These models comprise the formal foundations to specify and run a DEVS simulation [Zeigler et al. 2000]. An atomic DEVS model is defined as a 7-tuple M = $<$ X,Y,S,ta,δ_{ext}, δ_{int}, $\lambda >$

[8] SosADL is an architectural description language specially created for SoS domain.

[9] Details about the model transformation and how SosADL and DEVS models are mapped between them are not the focus of this article and can be found in [Graciano Neto et al. 2018c].

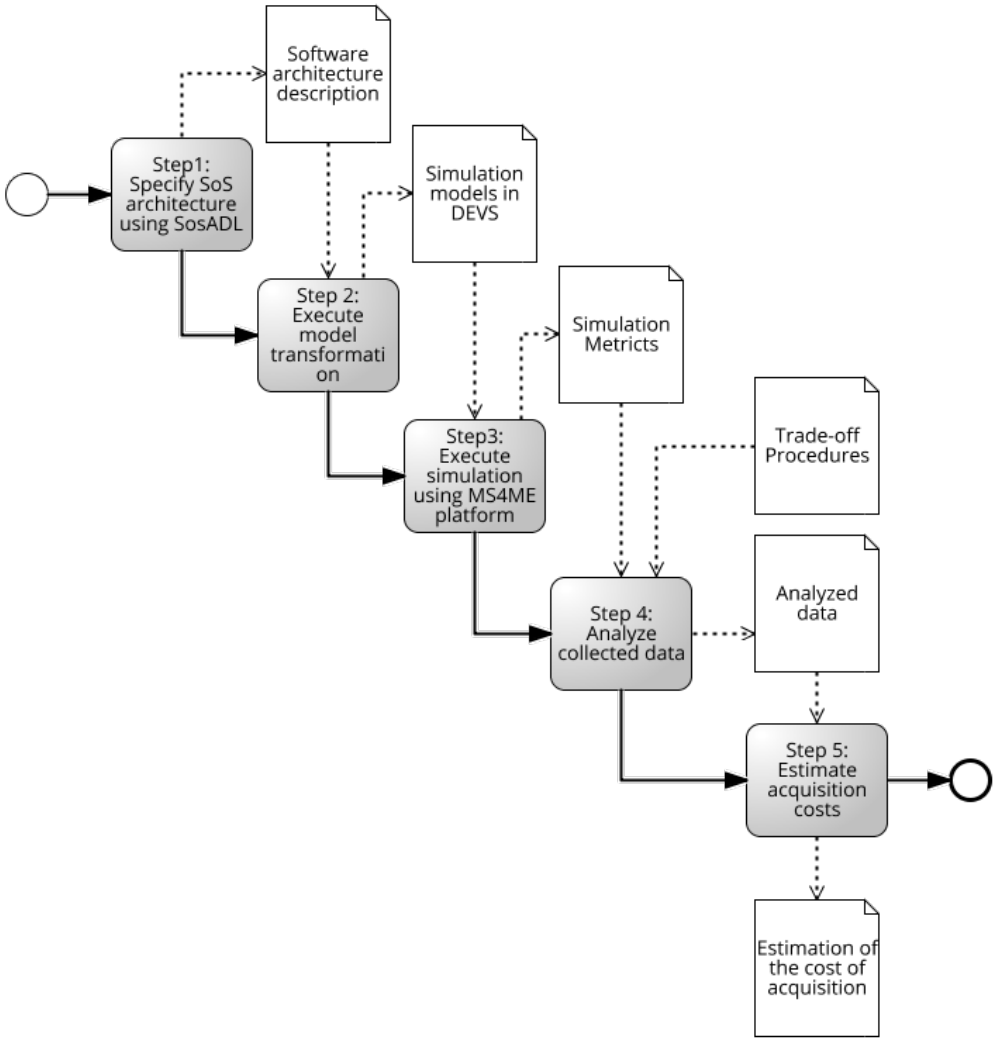

Figure 1. Overall Structure of S.O.B. Method.

where:

- X is the set of input events;
- Y is the set of output events;
- S is the set of sequential states (or also called the set of partial states);
- $s_0 \in S$ is the initial state;
- $ta : S \to T^{\infty} ta : S \to T^{\infty}$ is the time advance function which is used to determine the lifespan of a state;
- $$\delta_{ext} : Q \times X \to S \delta_{ext} : Q \times X \to S$$
  is the external transition function which defines how an input event changes a state of the system, where $Q = \{(s, t_e) | s \in S, t_e \in (T \cap [0, ta(s)])\} Q = \{(s, t_e) | s \in S, t_e \in (T \cap [0, ta(s)])\}$ is the set of total states, and $t_e t_e$ is the elapsed time since the last event;

- $\delta_{int} : S \rightarrow S$ is the internal transition function which defines how a state of the system changes internally (when the elapsed time reaches to the lifetime of the state);
- $\lambda : S \rightarrow Y^{\phi} \lambda : S \rightarrow Y^{\phi}$ is the output function where $Y^{\phi} = Y \cup \{\phi\} Y^{\phi} = Y \cup \{\phi\}$ and $\phi \notin Y \phi \notin Y$ is a silent event or an unobserved event. This function defines how a state of the system generates an output event (when the elapsed time reaches to the lifetime of the state);

A coupled DEVS model is defined as an 8-tuple

$N =< X, Y, D, \{M_i\}, C_{xx}, C_{yx}, C_{yy}, Select >$

where:

- X is the set of input events;
- Y is the set of output events;
- D is the name set of sub-components;
- $\{M_i\}\{M_i\}$ is the set of sub-components where for each $i \in D, M_i i \in D, M_i$ can be either an atomic DEVS model or a coupled DEVS model.
- $C_{xx} \subseteq X \times \bigcup_{i \in D} X_i C_{xx} \subseteq X \times \bigcup_{i \in D} X_i$ is the set of external input couplings;
- $C_{yx} \subseteq \bigcup_{i \in D} Y_i \times \bigcup_{i \in D} X_i C_{yx} \subseteq \bigcup_{i \in D} Y_i \times \bigcup_{i \in D} X_i$ is the set of internal couplings;
- $$C_{yy} : \bigcup_{i \in D} Y_i \rightarrow Y^{\phi} C_{yy} : \bigcup_{i \in D} Y_i \rightarrow Y^{\phi}$$ is the external output coupling function;
- $Select : 2^D \rightarrow D$ is the tie-breaking function which defines how to select the event from the set of simultaneous events;

Step 3. Simulation deployment and execution using MS4ME platform. DEVS models produced in Step 2 are deployed in MS4ME simulation environment. For this, `.dnl` les comprise the representation of structure and behavior of the individual systems in the form of DEVS *atomic models*. In turn, `.ses` model represents the DEVS *coupled model*, which captures how constituents interoperate, the structure of the entire SoS software architecture, and the emerging behavior as a result of data exchange among constituents. `.dnl` files are placed in the `Atomic models` directory of the Simulation Project in MS4ME, whilst the `.ses` model are deployed in the respective directory for coupled models of the Simulation project. Such Eclipse-based environment enables: (i) visualization of messages exchanged among constituents during SoS execution; (ii) dynamic architecture simulation; and (iii) measurement of pre-established metrics related to quality attributes.

Step 4. Analysis of collected data. Once the simulation is executed, a log of outputs is stored in a .CSV le that can be opened in a spreadsheet software so data can then be analyzed. It is possible to analyze values delivered by coalitions through a trade-off procedure, supporting the decision of the coalition that offers better combination between cost and benets; and

Step 5. Estimation of acquisition costs. Using results of Step 4 and considering the delivered results and total acquisition cost for each coalition, it is then possible to select the best option of coalition, considering the available budget and required quality. A table of prices can be used to estimate (with precision) the cost of acquisition for that set of constituents.

4. Evaluation

This section reports the Smart Building case study used to evaluate the S.O.B. method. Case study is a empirical, exploratory and hybrid qualitative-quantitative method to provide evidence about a research subject [Yin 2017]. This case study was conducted according to the following steps [Runeson and Höst 2009]: (i) case study design (preparation and planning for data collection); (ii) execution (collection of evidence); (iii) analysis of collected data; and (iv) reporting.

4.1 Study Protocol

4.1.1 Context of Study

Smart buildings provide important services to their residents and visitors, using data gathered by sensors and Internet of Things (IoT) systems to improve their experience and offer more elaborated behaviors, such as temperature and light control according to the data sensed. These sensors and systems refer to constituent systems of a Smart Building SoS (SBS), which was inspired in previous studies [Gassara et al. 2017, Manzano et al. 2018]. We emphasize that this work does not include proprietary systems centralized in a single controller. Although there are suppliers that can group a set of sensors and other components in a controller, and from this controller have access through interface and/or programming, this work is based on the individual use of components with access to open systems and the absence of a central controller. We also remark that we adopt the premise 'the constituents are interoperable', i.e., although each sensor has a set of configurations that can work properly or not with another set of configurations of other sensors, we do not consider the potential of interoperability between them and assume that they successfully interoperate.

Figure 2 displays a conceptual model of SBS with its constituent systems through a Block Definition Diagram of SysML; whilst Figure 3 illustrates a small-scale conception of the architectural elements of the SBS. Each block represents a different system. The scenario of this case study consists of a SBS composed of other three SoS: (i) a Fire System responsible for controlling fire sprinklers and issuing alarm of the building areas, e.g., corridors, rooms, and halls; (ii) a Lightning System that aims at controlling the light of building areas and the light intensity by means of the Lighting System Control Unities (LSCU); and (iii) a Room System that comprises private and self-contained environments composed of smoke sensors, temperature sensors, and presence sensors. These three SoS are managed by the Smart Building Control Unity (SBCU).

The missions defined to the SBS are threefold: (i) *light management*; (ii) *temperature control*; and (iii) *fire alarm management*. The light and presence sensors in combination with the smart lamps are installed in areas of the building. They interact with LSCU to activate or deactivate lamps in the *light management* mission.

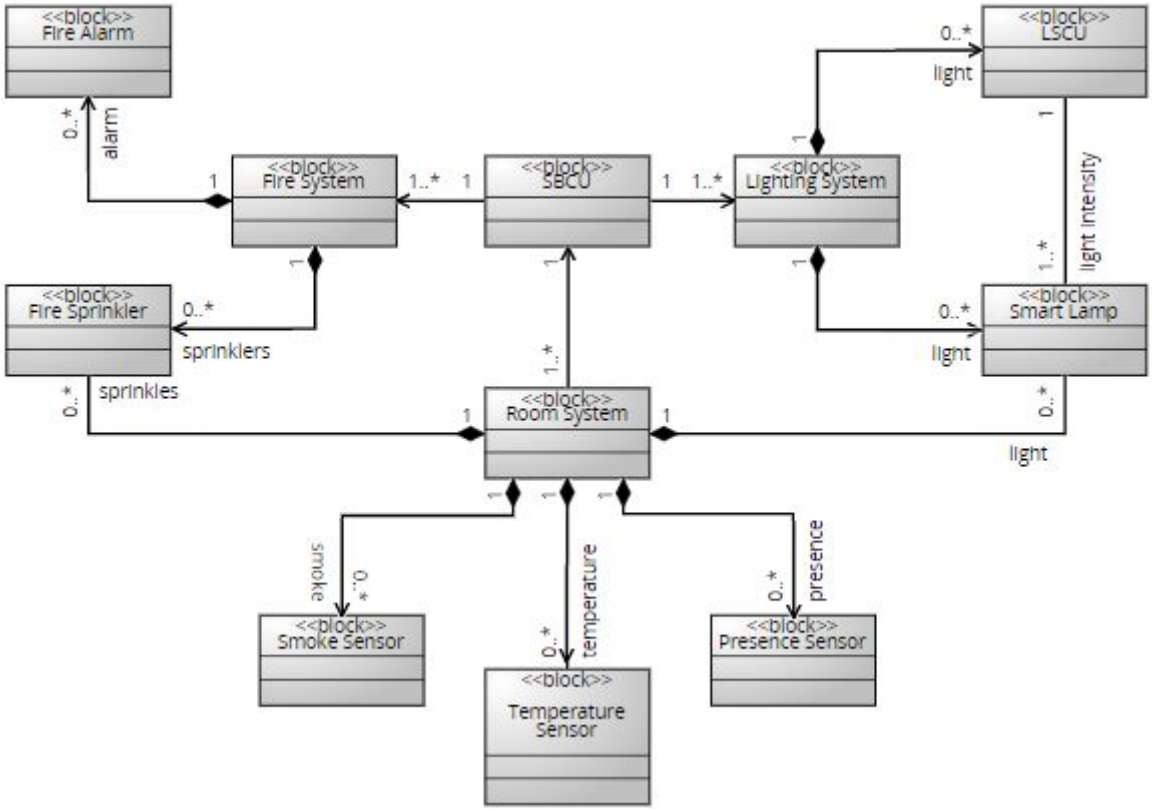

Figure 2. Smart Building SoS

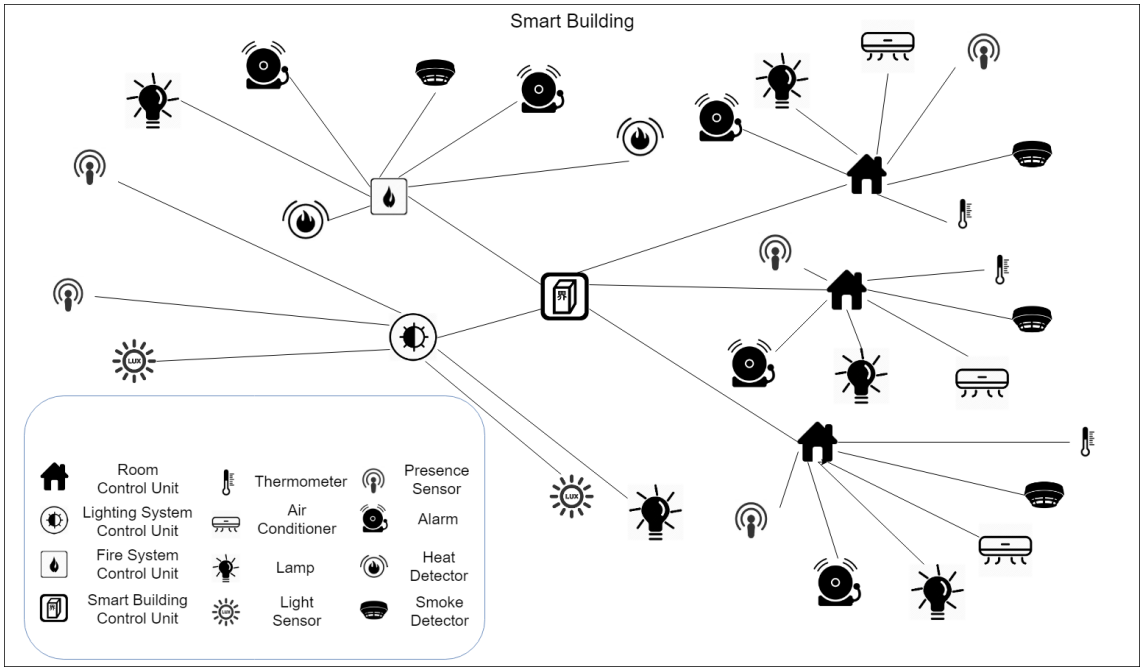

Figure 3. A small-scale illustration of the architecture of a Smart Building SoS.

Similarly, thermometers, presence sensors, and air conditioners, which may be installed in the rooms of the building, cooperate to provide an ideal room temperature previously configured by residents and visitors, i.e., the *temperature control* mission. Finally, the *fire alarm management* mission pulls together smoke sensors and heat sensors to detect a fire and notify the FSCU, which in turn may trigger alarms to inform people and activate fire sprinklers for putting out the fire.

4.1.2 Case Study Goals and Scope

We used the Goal-Question-Metric approach to establish our research [Basili et al. 1992]. On the basis of the mentioned SBS with its constituent systems (which are also SoS) and three missions, the goal of this case study is:
Goal: To assess whether the S.O.B. method supports a SoS architect to predict the acquisition cost for a SoS considering different coalitions (i.e., architectural arrangements) that can emerge due to different constituent suppliers (and costs) and the resulting quality.
Rationale. Based on simulations of SoS software architectures at design-time, S.O.B. method was designed to allow the architects to predict the cost of acquisition of constituents considering their contribution to the mission accomplishment, their acquisition cost, and some attributes of the different coalitions.
Then, we established the following research question and their respective metrics:
Question. *Can S.O.B. method support the prediction of acquisition costs for a SoS, offering options of coalitions to allow decision makers to decide on the suppliers and the number of constituents they want to acquire according to budget and intended quality?*
Rationale. This question investigates whether S.O.B. method can support the SoS analysis according to its functional suitability, and can decide better coalitions, i.e., those that offer better results. Considering an architectural plan already established for a smart building, the aims of this study is to reveal at design-time for the user: (i) whether different numbers of constituents could provide better results from others; and (ii) whether different suppliers could provide most valuable results than others in a same coalition.
Metrics: To assess these parameters, we adhere to ISO 25010 standard [ISO/IEC 2011], and evaluate the functional suitability according to two inherent sub-attributes, which are rewritten as follows:

- **Functional Completeness (FCom).** Degree to which the set of functions covers all the specified tasks and user objectives, i.e., considering the set of the three pre-established mission assigned to the SBS, how many of them are effectively achieved by the SoS? This metric assesses this number; and
- **Functional Correctness (FCorr).** Degree (percentage) to which the set of pre-established missions are achieved by the SoS, i.e., considering all the stimuli that is given to the SoS, which is the percentage that goal is accomplished? For instance, regarding the mission *fire alarm*, for all the stimuli that are delivered for the constituents, how many times are the fire alarms correctly triggered (and correctly non-triggered), and how many

times are they not? We intend to analyze if the variance in the number of constituents also varies the results according to this metric and, if yes, which one offers better results.

Rationale. Essentially, architectural analysis activities have an indissociable nature with quality attributes. In particular, the quality of a SoS is primarily related to the accuracy of its operation, that is, the percentage of correctness with which constituents collect data from the environment and react to it to culminate in a greater precision of operation of the whole SoS. Simulation models allow to analyze the SoS behavior and the effect of the individual contribution of the constituents on SoS as a whole. In this sense, an appropriate quality attribute to be analyzed (related to the quality with which the entire SoS fulfills its mission) was the functional suitability and its sub-attributes.

4.1.3 Research Instruments

We adopted Eclipse Modeling Framework (EMF) as the platform to develop SosADL models based on Xtext framework[10]. Xtend[11] is the transformation language, MS4ME[12] is the simulation platform, and DEVS (in particular, a DEVS dialect called DEVSNL) is the formalism used to specify the generated simulation models.

4.1.4 Models and Data Preparation

We adopted pre-existing models of a smart building based on modeling done in previous studies [Manzano et al. 2018, Gassara et al. 2017]. Three different versions were created for each constituent type, so that each version represents a different supplier of that type of constituent. Analogously, datasets were built to feed the simulation through the stimuli generators, creating a different set for each supplier of each constituent. Artificial errors were included in the datasets of some suppliers to imitate possible low quality, including "Presence Not Detected" by light sensors, "Fire Alarm Not Launched" by fire alarms, or "Temperature wrongly read" by termometers. The aim was to observe the impact of the errors in the final behavior of the SoS, and, as an outcome, to allow the analysis of which supplier would be better to acquire, considering the results obtained and the prices of each coalition.

To simulate a Smart Building, we built a realistic dataset to feed the simulation. The generated dataset was composed of data that represent 10 days. To stimulate Light Sensors, we used a type of data known as Lux (lx), which is the total luminous flux incident on a surface per unit area (illuminance). Such data were generated in a range between 0.1 lx (at night) and 10,000 lx (in broad daylight). The data received from Light Sensors by the BCU are used to turn on external lamps if the illumination is less than 100. To feed the Presence Sensor, data were

[10]https://www.eclipse.org/Xtext/

[11]https://www.eclipse.org/xtend/

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

generated randomly between 10 and 60 presences per sensor per day. This data was used by BCU to switch the lamps on in the Presence Sensor area. The data used to stimulate the Smoke Sensors consists of binary values (one, for smoke detected; and zero, for smoke not detected), and 10 fires were chosen in a random area. This data is sent to an FSCU or RCU. If the data value is 1, the alarm is triggered and the fire sprinklers are activated in the detected smoke area. Finally, the data generated for the thermometers were generated between 10ºC and 30ºC. This data is used by the RCU to turn on air conditioners if a person is detected in the area and the temperature is higher than a set temperature.

To get a more precise perception of quality, we opted to test one supplier of each type of constituent per coalition and observe how they presented different results. Thus, five different coalitions were created, so that from one to the other only the supplier and the number of constituents were varied; and also coalitions in which many constituents of a supplier were used only to try to compensate for the fact that quality and price are lower. The intention of the study was not to be exhaustive since the amount of possible combinations of constituents and suppliers is enormous. So, the idea is to allow some parameters to be analyzed by considering some possible combinations to choose a more expensive but with better quality, or cheaper and lower quality.

Two different suppliers were then determined for each constituent, one cheaper and one more expensive. Two coalition versions were created for each supplier, one with few cheap constituents, another with many cheap constituents; one with few expensive constituents, another version with many expensive constituents; and a last coalition with a mix between cheap and expensive ones to observe how the SoS behaved, as shown in Table 2. Table 4 shows the different prices for different sensors that work on Raspberry Pi single-board computer. All prices were collected in US$ on December 2nd, 2018. Table 3 presents the architectural arrangements considering the aforementioned rationale for each coalition.

	Expensive Constituents	**Cheap Constituents**
Coalition 1	None	Few
Coalition 2	None	Many
Coalition 3	Few	None
Coalition 4	Many	None
Coalition 5	Few	Few

Table 2. Description of coalitions elaboration for the study.

Moreover, to better represent constituents lower and higher quality, the simulation models of the constituents were elaborated according to a premise that cheaper constituents had less precision in their operation than expensive ones. Then, these models were designed to represent this fact with each one of them presenting a probability to fail, i.e., each version of each constituent (cheaper or expensive) was equipped with a probability (at simulation model level) of exhibiting false positives and false negatives regarding their expected functionalities, such as smoke detection or presence detection. Table 4 illustrates an excerpt of the rationale for some of the constituents. For instance, the Smoke Sensor with High price (Line

3) has only 1% of probability to exhibit a false negative. This means that from all the data received, there is only 1% of chance that a negative is wrongly detected by it.

Constituent		False Positive (%)	False Negative (%)
Smoke Sensor	Low	5%	2%
	High	2%	1%
Heat Sensor	Low	5%	2%
	High	2%	1%
Pesence Sensor	Low	10%	30%
	High	4%	7%

Table 3. Probabilities of false positive and false negatives in each type of Constituent.

Product	Cost	Individual Price (US$/unit)
Smoke Sensor	Low	15.99
	High	45.94
Heat Sensor	Low	12.38
	High	38.15
Smoke Alarm	Low	17.99
	High	42.94
Fire Sprinkler	Low	21.99
	High	60.94
Fire System Control Unit	Low	25.00
	High	34.95
Light Sensor	Low	15.50
	High	43.54
Pesence Sensor	Low	11.78
	High	36.87
Smart Lamp	Low	9.75
	High	12.49
Lighting System Control Unit	Low	25.00
	High	34.95
Thermometers	Low	12.00
	High	37.52
Air Conditioner	Low	117.00
	High	129.00
Room Control Unit	Low	34.95
	High	129.00

Table 4. Prices for each supplier of each type of sensor.

4.1.5 Analysis Procedures of Collected Data

According to Runeson and Höst [Runeson and Höst 2009], data analysis procedures can be quantitative or qualitative. For the former, the analysis is typically based on descriptive statistics and development of predictive models. All of these

	Coalitions					
Constituents	**Price**	**1**	**2**	**3**	**4**	**5**
Smoke Sensor	Low	25	45	0	0	25
	High	0	0	25	45	25
Heat Sensor	Low	5	15	0	0	5
	High	0	0	5	15	5
Smoke Alarm	Low	25	45	0	0	25
	High	0	0	25	45	25
Fire Sprinkler	Low	30	35	0	0	30
	High	0	0	30	35	30
Fire System Control Unit	Low	2	2	0	0	2
	High	0	0	2	2	2
Light Sensor	Low	25	45	0	0	25
	High	0	0	25	45	25
Pesence Sensor	Low	25	45	0	0	25
	High	0	0	25	45	25
Smart Lamp	Low	35	55	0	0	35
	High	0	0	35	55	35
Lighting System Control Unit	Low	1	3	0	0	1
	High	0	0	1	3	1
Thermometers	Low	25	45	0	0	25
	High	0	0	25	45	25
Air Conditioner	Low	25	50	0	0	25
	High	0	0	25	50	25
Room Control Unities	Low	15	25	0	0	15
	High	0	0	15	25	15
TOTAL		238	410	238	410	476

Table 5. Coalition architectural arrangements.

activities are relevant in case study research. For latter, the aim is to derive conclusions from data, keeping a clear chain of evidence [Seaman 1999]. In our study, we adopt a quantitative approach to measure the functional suitability of a multiple coalition of a SBS architecture based on simulations of its architectural specification. As discussed, this quality attribute (functional suitability) is analyzed according to its functional completeness (FCom) and functional correctness (FCorr), which are given by numbers representing respectively the number of missions that are effectively achieved and the percentage of correct behaviors performed by the SoS (compared to the expected behaviors).
For cost estimation purposes, the following function was created:

$$C = \sum_{i=1}^{12} n_i * p_i$$

where *C* is the total acquisition cost estimated for each coalition. *C* is the sum of the number of each type of constituent that will compose the coalition multiplied by its respective price. n_i is the number of the constituent *i* that will compose that coalition. For instance, from Table 5, *n* is 25 for constituent 1 (Smoke Sensor). In turn, p_i is the price of each of the twelve different types of constituents that are intended to compose each coalition analyzed, as shown in Table 4. The price is

then multiplied by the number of constituents of each supplier for that coalition. For instance, *$15.99* is the value of p_i for coalition 1, which is multiplied by 25, i.e., the number of low cost smoke sensors used in that coalition.

4.2 Reporting

We report our results based on the steps systematically followed to achieve the derivation of the stimuli generators for SBS constituents. Supplementary material is available at an external link[13], such as the complete SoS architecture specification documented in SosADL and the DEVS code that were produced via model transformations. We detail the procedures as follows.

Step 1. Specification of a SoS architecture using SosADL. The Smart Building specification was conceived using SosADL. Models were elaborated by one SosADL expert during almost two months of work. Refinements were performed to reinforce the precision of the software architectural description. Five different coalitions were modeled using the general organization of a Smart Building illustrated in Figure 2 and according to the arrangements listed in Table 3. For each type of constituents previously mentioned, two different versions were created representing two different suppliers of each constituent. Likewise, the data that were artificially created to represent the stimuli to be given to each system of that SoS were modified, creating a different set for each constituent supplier. Artificial errors were inserted in the datasets to imitate malfunction and low quality in the low price constituents. Examples of failures include "Presence Not Detected", "Fire Alarm Not Released", and "Temperature read wrong". Those oscillations in the constituent's behavior enabled to observe the consequences of low-quality constituents in the final behavior of the SoS. Moreover, such analysis was also drawn for each different coalition, determining the combination of constituents that was more valuable to acquire, considering the results obtained and the prices of each coalition. At the end of the process, the S.O.B. method provided us the result of the percentage of times in which each coalition was able to accomplish each of the missions, and whether all three missions were met in each coalition.

Such specification was validated by a peer-review procedure composed of other SoS experts. Two SoS experts were consulted on Smart Building modeling and simulation results. These experts had knowledge of the domain, and were not involved in any of the stages of this work and nor in the co-authoring of this article. Both experts agreed that the model satisfied that domain and that the results obtained were in conformance with what was expected. A verification was performed by inspection of the model and the obtained results.

Step 2. Model transformation execution. After the accomplishment of the first step, the automatic derivation step was conducted. The software architectural description produced in Step 1 was used as input for this step and processed by the model transformation script. SosADL models were analyzed by the transformation algorithm, and equivalent DEVS models were generated as the outcome of

[13]http://www.inf.ufg.br/~valdemarneto/projects/sosmethod.html

the model transformation. Those models were deployed in the MS4ME environment to conclude Step 2 and make possible Step 3. Besides producing simulation models for each one of the constituents being represented in the SoS architecture description documented in SosADL, the model transformation also produced an artificial entity called *stimuli generator* [Graciano Neto et al. 2017b], which are responsible by representing the surrounding environment in which the constituents are deployed, continuously delivering stimuli to feed the simulation. The stimuli are obtained from text files, which store data that represent environmental stimuli that can be sensed by the constituent systems, such as light, temperature, and presence. One stimuli generator was produced for each one of the constituents involved in the SoS. All the stimuli generators were deployed together with other models representing constituents.

Step 3. Simulation deployment and execution using MS4ME platform After deployed, the simulation was prepared to run. The realistic data produced was stored in text files, and the stimuli generators were connected to them to feed the simulation. The simulation was initiated with a first architectural configuration, as described by Coalition 1 in Table 3, i.e., 25 low price constituents of each type (Smoke sensors, heat sensors, and others), one lighting system control unit, and 15 room control unities. Data representing two whole days of sensing were used for each coalition. As the data prepared to the first coalition were totally consumed by the simulation, an artificial entity called Dynamic Reconfiguration Controller (DRC) was triggered to perform architectural changes and create a new architectural version of that SoS. At that moment, Coalition 2 was running with data for two days as well. This process was systematically repeated, simulating 10 days of the stimuli samples and covering the five pre-defined different coalitions. The simulation took 10 hours and 27 minutes running on a Intel(R) Xeon(R) CPU E5-2620 v3, with 30 GB RAM, 2 TB HD, running on Ubuntu Server 16.04.3 LTS.

Step 4. Analysis of collected data. Figure 4 shows the percentage of mission accomplishment for each one of the five different coalitions and results of the three pre-established missions. The X axis represents the coalitions (from 1 to 5), whilst the Y axis represents the percentage of fulfillment of the pre-established missions: (i) *light management*; (ii) *temperature control*; and (iii) *fire alarm management*. After the simulation finishes its execution in Step 3, the data are stored in text files and are analyzed comparing the total accomplishments of each mission in relation to the total number of inputs there were sent to each of the constituent systems. Once those data are available in regards to the percentage of accomplishment of the set of mission by each coalition and the respective prices of acquisition for each different SoS architectural arrangement, the SoS architect can draw the following analysis: by analyzing Table 5, it is possible to check that Coalition 3 exhibits the highest scores (around 96% percent of missions accomplishment), despite its price being 12K US dollars. Coalition 4 is also a good option from the functional completeness point of view; however, it is almost two times more expensive. Coalition 5 is also more expensive than Coalition 3 and is less successful. Coalition 1 and Coalition 2 are both cheaper than Coalition 3; however, their performance in mission accomplishment is remarkably lower than Coalition

3. Hence, Coalition 3 is the best option.

By analyzing the plotted data, the conclusion is that the best coalition was the third one (Coalition 3), which involved a small number of more expensive constituents. Observing the plot, it is possible to conclude an increase in the number of constituents decreases the effectiveness of the SoS to achieve the missions. This happens because all constituents were connected to a Control Unit. The constituents of the Fire System connected to the Fire System Control Unit, the Room with the Room Control Unit and the Lighting System with the LSCU. Hence, since we had a point where the data was received and processed, this point might be processing/receiving another data while other data arrived, causing data losses, consequently affecting the number of data received and the percentage of missions effectively accomplished.

Coalition	***Fire Control (%)***	***Light Management (%)***	***Air Conditioning (%)***	***Average of Mission Accomplishment (%)***	***Total Cost per coalition ($)***
1	73.12	65.94	74.03	71.03	6,418.39
2	65.48	57.21	66.25	62.98	11,636.67
3	98.58	96.91	94.71	96.73	12,891.22
4	92.15	91.61	91.18	91.646	22,548.36
5	84.12	75.65	82.96	80.91	19,309.61

Table 6. Percentage of missions accomplishment per coalition and respective acquisition prices.

According to the pre-established metrics, the ***functional completeness (FCom)*** of the SBS, i.e., the proportion of the missions among the pre-established mission that were accordingly achieved, was 100% (three out of three pre-established missions were effectively achieved by the SoS). In turn, Table 6 summarizes the percentage of accomplishment of each mission achieved by each one of the coalitions. The maximum percentage of missions accomplishment was achieved by the Coalition 3 for the mission *fire control* (98.58%). This happened because the smaller number of errors performed by the individual constituents related to their functionalities (sense the temperature, presences, or light) resulted in a smaller total number of non-accomplished missions (only 1.42% were not achieved due to individual malfunctions of the inner constituents). Therefore, the best average of percentage of missions accomplishment was achieved by the coalition with 238 expensive constituents.

Figure 5 shows an average of the percentage of missions that was achieved during the SoS simulation and the respective total acquisition prices for each coalition. This figure summarizes the data presented in column “Average of Mission Accomplishment” (Table 6). Considering the ***functional correctness (FCorr)***, which comprises the extension to which the set of pre-established missions are correctly achieved by the SoS, we observe that the minimum correctness achieved by the SoS was 57.21% of the light management system correctly sensing presences; and that the third coalition exhibited the best quality considering the pre-established

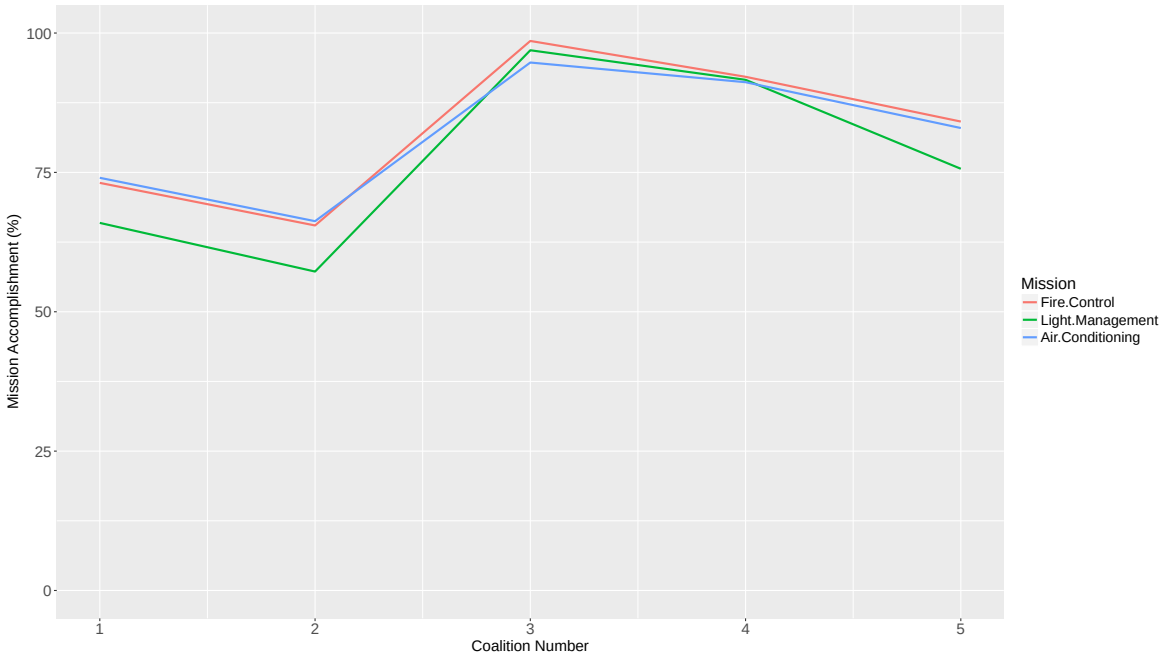

Figure 4. Percentage of achievement of three missions for each different coalition assumed by the Smart Building SoS.

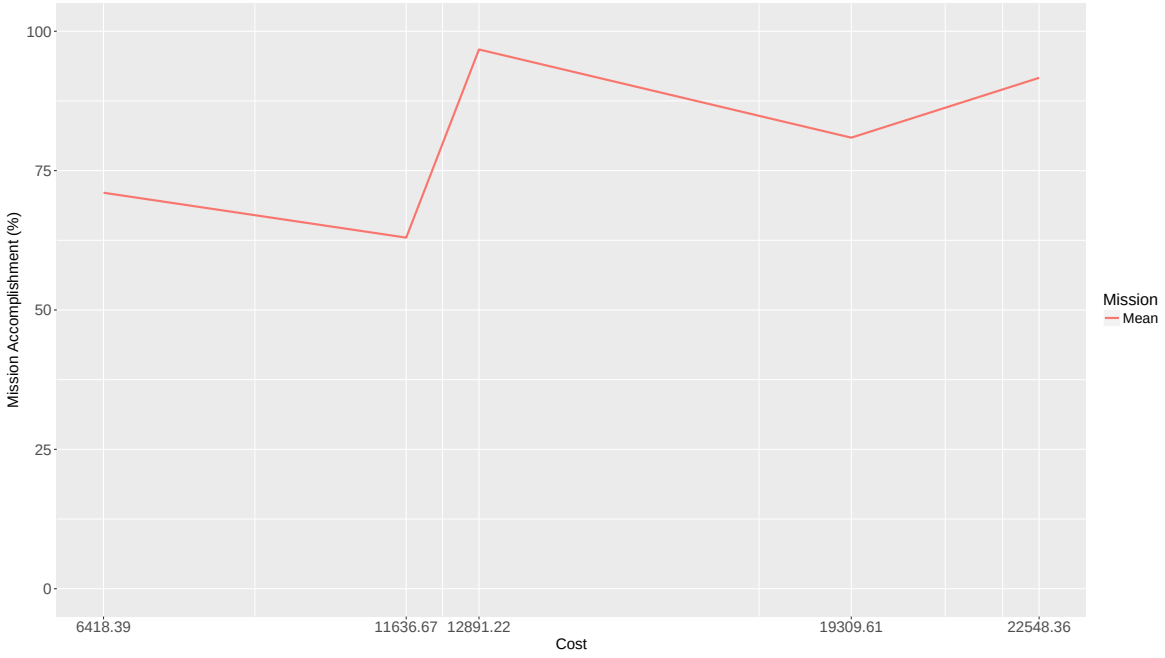

Figure 5. Average of missions accomplishment for each coalition and respective prices of each coalition.

metrics and attributes, reaching an average of 96.73% of missions correctly accomplished.

Step 5. Estimation of acquisition costs. The cost estimation was calculated

using the prices displayed in Table 4 and multiplying it by the number of constituents of each supplier for each different coalition, as shown in Table 3. Table 6 presents data related to the total cost associated to the acquisition of each of the different coalitions. We observe that the best benefit-cost ratio is obtained by the Coalition 3, which exhibits an average price considering the most expensive and the cheapest, while still offering better results considering the pre-established metrics.

Then, we can answer the raised research question: *Can S.O.B. method support the prediction of acquisition costs for a SoS, offering options of coalitions to allow decision makers to decide on the suppliers and the number of constituents they want to acquire according to budget and intended quality?* The answer is Yes. For this case study illustrated and discussed herein, the S.O.B. method supported the analysis of different coalitions considering a set of pre-established quality parameters, besides allowing an architect to observe different combinations of constituents and suppliers, and to predict the acquisition prices of each one of those coalitions.

4.3 Discussion

S.O.B. method allows an architect to predict, at design-time, the effectiveness achieved by multiple coalitions to accomplish a set of SoS missions. Such analysis is performed according to specific quality attribute (functional suitability in our case study), also supporting the prediction of the total acquisition costs of each coalition. Information delivered by S.O.B. method enables a trade-off analysis, considering functional properties of a SoS (i.e., the missions and their accomplishment) and their respective non-functional properties as well (including price and functional suitability). We claim that, using S.O.B. method, cost-benefit ratio can be identified, supporting decision makers to decide which constituents could be acquired to obtain a given quality, but respecting economic constraints.

The software architecture of a system composed of multiple constituents consists of the software of each of these constituents added to the elements that allow the interoperability between them. Once SoS are mission-oriented, that is, they are developed to accomplish a set of behaviors, bringing together systems that offer the necessary functionalities to accomplish the established missions is a prime task. However, such systems may have different costs and quality levels. As such, acquisition costs directly interfere in the acquirer's ability to perform the acquisition and in the quality with which the mentioned functionalities are delivered. Then, it is necessary to carry out a study that provides, at project time, a preview of the possible costs and combinations of constituents based in the required functionalities and in the expected quality. To do so, a simulation study is carried out to provide the relationship between the prices of constituents and quality delivered so that the acquirer decides on which coalition s/he should obtain based on the entire software system that emerge from constituents interoperability. Therefore, the relationship between product, architecture and software in our method of acquiring constituents is given by an analysis, at project time and through simulation, of the potential of delivering quality functionalities by different combinations of

constituents, culminating in different combinations of functionalities.

Our method covers essential characteristics of SoS that are not covered by other studies. Other studies such as [Burton et al. 2014] have focused on optimization problems to find, within the expected spectrum of constituent capabilities, the minimum set of constituents, without a thorough analysis of functional suitability or other quality attributes. Our method analyzes results delivered by different coalitions according to a set of metrics (pre-defined in the context of the SoS architectural analysis approach), allowing a trade-off between quality and cost of the SoS.

This work also contributes to previous works on the role of architects in software ecosystems [Weinreich and Groher 2016, Amorim et al. 2017]. Within such ecosystems, architects are responsible by defining better strategies to the software products as they know the customers' needs and priorities. Results obtained using our method could provide them valuable metrics, as well as the arrangements of SoS that could support decision-making task. In other words, they can make decisions considering not only customers' needs but also a lower cost and better quality of systems. Besides that, the model transformation mechanism is another contribution provided by our work. It was used to produce the simulation models for the case study presented in this article as well as in other two different application domains. Hence, providing this mechanism fosters reuse in SoS, since it could be potentially reused in many other domains.

S.O.B. method also contributes to the Model-Based Engineering, which together with its related methods, have been recognized for the SoS development [Graciano Neto et al. 2018b, Zeigler et al. 2018]. Model-Based Engineering (MBE) is the practice of systematically using models during an engineering activity [Agner et al. 2013]. By providing an infrastructure that automatically obtains an executable model from a static SoS architectural specification, S.O.B. method contributes to facilitate the systematic use of models during SoS Engineering. Hence, we provide not only an approach to predict the cost in constituents acquisition processes, but also a model-based approach that prescribes the use of executable models (simulation models) to support a more precise prediction of their costs and the impact on the resulting quality. By contributing to the systematic adoption of models for engineering activities, model-based SoS engineering is benefited by our approach. In particular, our method was built on previous advances and provides a model-driven approach to support at design-time the cost prediction in SoS constituents acquisition processes. This is valuable, contributing then to the SoS Software Engineering and could be extended to Systems-of-Information Systems (SoIS), one of the Grand Challenges for Information Systems in Brazil between 2016 and 2026 [Graciano Neto et al. 2017a].

MDE provides a means, through model transformation, to use models for representing how constituents should interoperate to accomplish missions to automatically generate configuration files, underlying middleware, and glue code to support constituents interoperability [Graciano Neto et al. 2014]. In S.O.B. method, MDE was used to automatically generate simulation models that can be used to predict the interoperability of SoS in a real environment, and adaptations may be

made to give even more accurate interoperability of simulation models.

4.4 Threats to Validity

Threats to validity can be of four types [Wohlin et al. 2000]: conclusion, internal, construction and external. Conclusion validity is concerned with the statistical relation between the initial data and the outcome. Internal validity are related to factors that affect the outcome. Construction validity concerns the extent to which measures accurately reflect the theoretical concepts they are intended to measure. External validity refers to the generalization of research findings [Neto and Conte 2013].

The statistical relations in our study were drawn based on percentages, i.e., the proportion of the missions that were accordingly and correctly accomplished in a total of intended missions while considering different SoS architectural arrangements. Since there is no hypothesis test in our study, we only glimpse threats to **conclusion validity** related to our premise that expensive suppliers exhibit a better quality, whilst cheaper constituents offer lower quality. This was an assumption specically established for this study that does not invalidate any of the obtained conclusions. However, supplementary methods should be developed/adopted to previously measure the quality of the functionalities provided by each supplier of a type of constituent to support the construction of a more precise stimuli set to feed the simulation.

In regards to **internal validity**, we identied three classes of threats: (i) transformation correctness; (ii) human failure during prices estimation; and (iii) choice of the best coalition. Firstly, the same model transformation has already been used to make dozens of transformations between SosADL and DEVS models for two different domains: smart cities and space. Therefore, this threat is relieved by the number of studies that have already used such transformation. In addition, although formal proofs of its correctness have not been conducted, it generates correctly specied simulations every time. Such a result is reliable because in the DEVS formalism a single erroneous instruction may make the simulation execution unfeasible, causing it to crash or even preventing its execution. From the point of view of human failure, there are some points in the process that are subject to failures, such as the observance and collection of metrics, as well as the choice of constituent prices. For this, a study was performed on a small scale and results indicated the feasibility of reproducing it on a larger scale. Moreover, to reduce such threats, for the future, an automated process could be adopted to avoid human failures. The last threat to internal validity is related to the fact that individual configurations of constituent systems are not considered, i.e., each sensor used in the constituents can have a set of configurations that can work properly or not with another set of configurations of other sensors. Since we assume that the constituents are fully interoperable, this variable is not considered and the outcome could be affected due to this assumption. Since the focus of this study was to evaluate the functional suitability, we abstracted the quality attribute of interoperability, and did not consider it for the purpose of this specific study. Forthcoming advances shall include interoperability as a factor. To relieve this threat, researchers that

will use the outcome of this study should include a risk factor about a possible incompatibility among different sensor configurations.

Considering **construction validity**, we draw our conclusions based on an approach that was systematically followed to automatically derive and run the simulation. Our metrics were dened using GQM technique. The research question (Can S.O.B. method reliably support the prediction of acquisition costs for a SoS, offering options of coalitions to allow decision makers to decide on the suppliers and the number of constituents they want to acquire according to budget and intended quality?) is aligned with the goal (To assess whether the S.O.B. method supports a SoS architect to predict the acquisition cost for a SoS considering different coalitions, i.e., architectural arrangements, that can emerge due to different constituent suppliers - and costs - and resulting quality) and the respective metrics dened are accordingly subject to measurement (Functional Completeness and Correctness). Results are provided and the process is repeatable and auditable. Then, we claim this threat is relieved by the rigour of the procedure followed to establish the research protocol.

One identified threat to **external validity** is related to the fact that we did not exhaustively combined a more diverse number of different coalitions, suppliers, or types of failures that were articially created. Hence, our conclusions are roughly based on the input parameters considered and this could affect the potential of generalization of our results. Although this is an important issue, the intention of the study was to assess if the method was well-succeeded to support cost prediction for different suppliers of a same type of constituents and considering a quality parameter. The method not only enabled what was planned, but also supported a cost-benet ratio analysis, what is valuable for SoS engineering. Hence, this threat is relieved.

5. Final Remarks

Cost is one of the primary drivers to decide whether to build a SoS from existing constituents or to create a new specialized system from scratch [Johnson 2015]. Moreover, cost is a relevant economic aspect of systems. In this scenario, the main contribution of this article is S.O.B. method that enables to evaluate different coalitions (arrangements of constituents that could possibly be part of a SoS) and provides support to make decisions about which constituents could be acquired to form a given SoS, considering also quality and acquision costs. S.O.B. Method extends Graciano Neto et al.'s method [Graciano Neto et al. 2018c] by adding a step that enables the SoS architect to estimate cost acquisition based on pre-established quality attributes. According to Gregor and Hevner's Knowledge Contribution Scheme [Gregor and Hevner 2013], S.O.B method can be acknowledged as an "Invention´´ because it comprises a new solution (i.e., a simulation-based method to predict the acquisition cost for constituents) for new problems (i.e., cost estimation of SoS). Furthermore, we also analyzed the contributions of our work by following Gregor and Hevner's distinction between *descriptive* (i.e., knowledge of natural phenomena) and *prescritive* (i.e., knowledge of human-built artifacts) contributions [Gregor and Hevner 2013]. Thereafter, our contributions are threefold. To start with, the S.O.B method established a way of systematically

estimating the cost of acquisition of constituent systems of a SoS. This represents a *prescriptive* contribution. Secondly, lessons were learned from a case study and then may assist practitioners in determining the overall architecture of their SoS at design-time. Finally, the case study itself also comprises a valuable contribution to support experts on the design of Smart Building SoS. These two later achievements are characterized as *descriptive* contributions.

After conducting case studies (one of them is detailed in this article), we concluded that the most expensive coalition (U$ 22,548.36) does not bring us the better quality (approximately 91% of SoS mission accomplishments), but a mixed coalition (expensive and cheaper constituents, totaling U$ 12,891.22) could achieve good quality (96.73% of missions accomplishment) and an acceptable cost (almost a half than the most expensive). Having such information, it is possible to anticipate which constituents are effectively necessary to build a SoS, and predict the budget necessary to acquire them. The acquisition and construction of a SoS also involve acquiring hardware in which software will be deployed with its specific capabilities to collaborate to an intended emergent behavior. In this article, we exploit: (i) the prediction of software architectures of a SoS at design-time; (ii) the prediction of different coalitions that a SoS could assume at run-time; and (iii) the results that each one of such coalitions yield to support to support cost prediction; and (iv) a prediction of the acquisition costs related to corresponding hardware necessary to support the existence of that SoS.

In particular, in this article we advanced our research by extending the previous version of S.O.B. method and covering a limitation that we had raised. In the preliminary version of S.O.B. method, we had not explored different constituents suppliers being benchmarked to support the selection of better coalitions. In this version, we exploited two different suppliers for each type of constituents. More suppliers will be also tested in future works. Another perspective of investigation is to comprise the prediction under the man-hour metric and function points. Other future works include: (i) comparison among coalitions through the substitution of constituents that offer the same capability for a better decision-making of different suppliers; (ii) adoption of co-simulation to accurately reproduce the scenarios required for other quality attributes such as security [Hachem et al. 2016]; and (iii) establishment of a mechanism for automation of the cost estimation through the integration of a simulator, a mechanism for querying and comparing market prices, and some model-checker mechanism to automatically deliver better coalitions, without the need to manually collect and analyze data; moreover, we consider that, for large volumes of data, we can apply search-based software engineering to support the selection of constituents from criteria related to technical and economic aspects of software; (iv) investigation of coverage, testing multiple architectural conformations that a SoS can assume, as well as multiple stimuli that can be received, resulting in a testing approach for SoS [de Oliveira Neves et al. 2018]; in addition, different numbers of constituents, different constituent suppliers, and multiple quality attributes also need to be taken into account; (v) use of real data instead of realistic data that also needs to be experimented to possibly provide more reliable results; and (vi) optimization models that can also be added to the

S.O.B. method, since an increase in the number of quality attributes becomes less trivial to perform a successful trade-off analysis of quality attributes.
Finally, we highlight the importance of the results achieved until now and the seminal nature of our solution for the SoS domain. We hope S.O.B. method can be adopted for constituents acquisition processes in Brazil and worldwide.

6. Acknowledgements

The authors would like to thank FAPESP (grants 2017/06195-9, 2017/17448-5, and 2018/21517-5). The 4th author thanks to UFMA through the research project PVCET200-2017. FEAH is gratefull to the financial support of National Council for Scientific and Technological Development (CNPq) (Grant no.437937/2018-6).

References

Acker, D. D. (1983). Defense systems acquisition review process: A history and evaluation. Technical report, Defense Systems Management Coll Fort Belvoir Va.

Agner, L. T. W., Soares, I. W., Stadzisz, P. C., and Simão, J. M. (2013). A Brazilian Survey on UML and Model-driven Practices for Embedded Software Development. *J. Syst. Software*, 86(4):997–1005.

Akintoye, A. and Fitzgerald, E. (2000). A survey of current cost estimating practices in the uk. *Construction Management & Economics*, 18(2):161–172.

Amorim, S. S., McGregor, J. D., de Almeida, E. S., and von Flach G. Chavez, C. (2017). The architect's role in software ecosystems health. In *WASHES*, pages 1–4.

Asiedu, Y. and Besant, R. (2000). Simulation-based cost estimation under economic uncertainty using kernel estimators. *International Journal of Production Research*, 38(9):2023–2035.

Axelsson, J. (2018). An initial analysis of operational emergent properties in a platooning system-of-systems. In *2018 Annual IEEE International Systems Conference (SysCon)*, pages 1–8.

Basili, V., Caldiera, G., and Rombach, H. D. (1992). Software modeling and measurement: the goal/question/metric paradigm. Technical Report CS-TR-2956, UMIACS-TR-92-96, University of Maryland, College Park, Maryland, USA.

Bass, L., Clements, P., and Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Boehm, B., Abts, C., and Chulani, S. (2000). Software development cost estimation approaches—a survey. *Annals of software engineering*, 10(1-4):177–205.

Burton, F. R., Paige, R. F., Poulding, S., and Smith, S. (2014). System of systems acquisition trade-offs. *Procedia Computer Science*, 28:11–18.

Burton et al. (2012). Solving acquisition problems using model-driven engineering. In *ECMFA*, volume 7349, pages 428–443, Lyngby, Denmark. Springer.

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

de Oliveira Neves, V., Bertolino, A., Angelis, G. D., and Garces, L. (2018). Do we need new strategies for testing systems-of-systems? In *6th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems, SESoS@ICSE 2018, Gothenburg, Sweden, May 29, 2018*, pages 29–32.

Gassara, A., Bouassida, I., and Jmaiel, M. (2017). A tool for modeling sos architectures using bigraphs. In *32nd Symposium on Applied Computing (SAC 2017)*, pages 1787–1792, Marrakech, Morocco. ACM.

Graciano Neto, V. V., Guessi, M., Oliveira, L. B. R., Oquendo, F., and Nakagawa, E. Y. (2014). Investigating the model-driven development for systems-of-systems. In *Proceedings of the 2014 European Conference on Software Architecture Workshops*, ECSAW '14, pages 22:1–22:8, Vienna, Austria. ACM.

Graciano Neto, V. V., Horita, F. E. A., Santos, R. P., Viana, D., and Kassab, M. (2018a). How much does it cost? a simulation-based method for cost prediction in systems-of-systems acquisition processes. In *WASHES*, pages 1–10, Natal, Brazil.

Graciano Neto, V. V., Manzano, W., Kassab, M., and Nakagawa, E. Y. (2018b). Model-based engineering & simulation of software-intensive systems-of-systems: experience report and lessons learned. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings, ECSA 2018, Madrid, Spain, September 24-28, 2018*, pages 27:1–27:7.

Graciano Neto, V. V., Oquendo, F., and Nakagawa, E. Y. (2017a). *Smart Systems-of-Information Systems: Foundations and an Assessment Model for Research Development*, pages 1–12. Brazilian Computer Society, Porto Alegre, Brazil.

Graciano Neto, V. V., Paes, C. E., Garcés, L., Guessi, M., Oquendo, F., and Nakagawa, E. Y. (2017b). Stimuli-SoS: A model-based approach to derive stimuli generators in simulations of software architectures of systems-of-systems. *Journal of the Brazilian Computer Society*, 23(1):13:1–13:22.

Graciano Neto, V. V., Rodriguez, L. M. G., Guessi, M., Paes, C., Manzano, W., Oquendo, F., and Nakagawa, E. Y. (2018c). ASAS: an approach to support simulation of smart systems. In *51st Hawaii International Conference on System Sciences - HICSS 2018*, pages 5777–5786, Hilton Waikoloa Village, Hawaii, USA. IEEE.

Gregor, S. and Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. *MIS quarterly*, pages 337–355.

Hachem, J. E., Pang, Z. Y., Chiprianov, V., Babar, A., and Aniorté, P. (2016). Model driven software security architecture of systems-of-systems. In *23rd Asia-Pacific Software Engineering Conference*, pages 89–96, Hamilton, New Zealand. IEEE.

ISO (2011). Iso/iec/ieee systems and software engineering – architecture description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pages 1–46.

ISO/IEC (2011). ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. Technical report.

Johnson, S. B. (2015). System health management. In Rainey, L. B. and Tolk, A., editors, *Modeling and Simulation Support for System of Systems Engineering Applications*, pages 131–144. Wiley, Hoboken, New Jersey, USA.

Lowe, P. N. and Chen, M. W. (2008). System of systems complexity: Modeling and simulation issues. In *Proceedings of the 2008 Summer Computer Simulation Conference*, SCSC '08, pages 36:1–36:10, Vista, CA. Society for Modeling & Simulation International.

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

Manzano, W., Graciano Neto, V. V., and Nakagawa, E. Y. (2018). Dynamic-SoS: An approach for the simulation of system-of-systems dynamic architectures. *The Computer Journal*, X:1–23. Accepted for Publication.

Moløkken-Østvold, K., Jørgensen, M., Tanilkan, S. S., Gallis, H., Lien, A. C., and Hove, S. (2004). A survey on software estimation in the norwegian industry. In *10th International Symposium on Software Metrics, 2004. Proceedings.*, pages 208–219. IEEE.

Neto, A. A. and Conte, T. (2013). A conceptual model to address threats to validity in controlled experiments. In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, EASE '13, pages 82–85, Porto de Galinhas, Brazil. ACM.

Neto, V. V. G., Horita, F. E. A., Cavalcante, E., Rohling, A. J., Hachem, J. E., Santos, D. S., and Nakagawa, E. Y. (2018). A study on goals specification for systems-of-information systems: Design principles and a conceptual model. In *Proceedings of the XIV Brazilian Symposium on Information Systems, SBSI 2018, Caxias do Sul, Brazil, June 04-08, 2018*, pages 21:1–21:8.

Nielsen, C. B., Larsen, P. G., Fitzgerald, J., Woodcock, J., and Peleska, J. (2015). Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions. *ACM Comput. Surv.*, 48(2):18:1–18:41.

Olagbemiro, A., Shing, M.-T., and Mun, J. (2009). Application of real options theory to software-intensive system acquisitions. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA DEPT OF COMPUTER SCIENCE.

Oquendo, F. (2016). Formally Describing the Software Architecture of Systems-of-Systems with SosADL. In *11th Annual System of Systems Engineering (SOSE 2016)*, pages 1–6, Kongsberg, Norway. IEEE.

Ricci, N., Rhodes, D. H., Ross, A. M., and Fitzgerald, M. E. (2013). Considering alternative strategies for value sustainment in systems-of-systems. In *2013 IEEE International Systems Conference (SysCon)*, pages 725–730.

Robbins, W., Lam, S., and Lalancette, C. (2005). Towards a collaborative engineering environment to support capability engineering. In *Proceedings of the 2005 INCOSE International Symposium*, pages 211–221, Rochester, NY, USA.

Rodriguez, L. M. G. and Nakagawa, E. Y. (2017). A process to establish, model and validate missions of systems-of-systems in reference architectures. In *SAC 2017*, pages 1765–1772, Marrakech, Morocco. ACM.

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Engg.*, 14(2):131–164.

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on software engineering*, 25(4):557–572.

Sharma, N., Bajpai, A., and Litoriya, M. R. (2012). A comparison of software cost estimation methods: A survey. *The International Journal of Computer Science and Applications (TIJCSA)*, 1(3).

Silva, E., Batista, T., and Cavalcante, E. (2015). A mission-oriented tool for system-of-systems modeling. In *SESoS*, pages 31–36, Florence, Italy. IEEE.

Takakuwa, S. (1997). The use of simulation in activity-based costing for flexible manufacturing systems. In *Winter Simulation Conference Proceedings,*, pages 793–800.

Urwin, E. N., Pilfold, S. A., and Henshaw, M. (2010). Through life capability management: benefits and behaviours. In *International Conference on Contemporary Ergonomics and Human Factors*, pages 153–162. CRC Press.

Weinreich, R. and Groher, I. (2016). The architect's role in practice: From decision maker to knowledge manager? *IEEE Software*, 33(6):63–69.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA.

Yang, D., Wang, Q., Li, M., Yang, Y., Ye, K., and Du, J. (2008). A survey on software cost estimation in the chinese software industry. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 253–262. ACM.

Yang, I.-T. (2005). Simulation-based estimation for correlated cost elements. *International Journal of Project Management*, 23(4):275 – 282.

Yin, R. K. (2017). *Case study research and applications: Design and methods*. Sage publications.

Zeigler, B. P., Kim, T. G., and Praehofer, H. (2000). *Theory of Modeling and Simulation*. Academic Press, Inc., Orlando, FL, USA, 2nd edition.

Zeigler, B. P., Mittal, S., and Traore, M. K. (2018). Mbse with/out simulation: State of the art and way forward. *Systems*, 6(4).

Zeigler, B. P., Sarjoughian, H. S., Duboz, R., and Souli, J.-C. (2012). *Guide to Modeling and Simulation of Systems of Systems*. Springer, Berlim, Germany.