2016 21st International Conference on Engineering of Complex Computer Systems

Bridging Missions and Architecture in Software-intensive Systems-of-Systems

Eduardo Silva, Everton Cavalcante, Thais Batista
Department of Informatics and Applied Mathematics
Federal University of Rio Grande do Norte
Natal, Brazil
eduardoafs@ppgsc.ufrn.br, {everton, thais}@dimap.ufrn.br

Flavio Oquendo
IRISA – UMR CNRS
Université Bretagne Sud
Vannes, France
flavio.oquendo@irisa.fr

***Abstract*—Missions represent a key concern in the development of Systems-of-Systems (SoS) since they are related to both capabilities of constituent systems and interactions among these systems that contribute to the accomplishment of global goals of the SoS. In a mission-oriented approach to design software-intensive SoS, the activity towards the concretization of the mission model is its refinement to an architectural model. This paper addresses this synergetic relationship between mission and architectural models. As main contribution, we introduce a model-based refinement process supported by model-to-model transformations intended to apply mission models represented in mKAOS, a language to model missions, for automatically generating architecture descriptions in SosADL, a formal language to describe SoS software architectures.**

Keywords—systems-of-systems; missions; software architecture; architecture description language; model refinement.

I. INTRODUCTION

A *System-of-Systems* (SoS) results from local interactions of multiple *constituent systems* that cooperate to form a larger, more complex system for accomplishing a given mission [8]. Each of these constituent systems has individual missions and can contribute to the accomplishment of the global mission of the SoS. The collaboration among such constituent systems enables an SoS to offer new capabilities that cannot be provided by any of these systems working as individual entities, the so-called *emergent behavior*. Besides emergent behavior, there are other intrinsic characteristics that make an SoS distinct from other distributed complex and large-scale systems: (i) the *operational* and *managerial independence* of constituent systems, which provide their own functionalities even when they do not cooperate within the scope of an SoS and can be managed independently from it, and (ii) the *evolutionary development* of the SoS, which may evolve over time to respond to changes on its operational environment, on the constituent systems, or on its own mission. Altogether, these characteristics have posed a set of challenges which made system engineering processes to be no longer suitable for developing these systems [2].

An important concern in the design of SoSs is the systematic modeling of both global and individual missions, as well as all relevant mission-related information. Missions play a key role in the SoS context since they define required capabilities of constituent systems and the interactions among these systems that lead to emergent behaviors towards the accomplishment of the global goals of the SoS. Therefore, *mission models* are the starting point for designing an SoS and are used as a basis of the whole evolutionary development process.

In a mission-oriented approach for designing software-intensive SoS, the next step towards the concretization of the mission model is its refinement to an *architectural model*, i.e., a model expressing the SoS software architecture.

The SoS software architecture is recognized as the key factor for achieving missions [7]. Therefore, mission models can be used as a basis for the further elaboration of architectural models by SoS software architects. Such a refinement allows specifying the SoS software architecture in compliance with the mission model, so that it is possible to establish traceability links between missions and architectural elements.

This paper concerns the synergetic relationship between mission and architectural models. Our proposal relies on mission models described in mKAOS, a pioneering language introduced in our previous work [16][17] aimed to support the specification of missions and the definition of relationships between such missions and other concerns of the SoS.

The main goal of mKAOS is to allow for a detailed modeling of missions in the SoS context and to enable stakeholders to identify or define specific elements for the SoS, e.g., constituent systems, required capabilities, and/or desired emergent behaviors.

On the other hand, the architectural representation is addressed by using SosADL [10][12], a formal, well-founded theoretically architecture description language (ADL) targeting the description of SoS software architectures under both structural and behavioral viewpoints while intending to cope with the features of software-intensive SoS.

More precisely, SosADL provides novel architectural concepts and the language constructs concretely embodying the SoS architectural concepts coping with the SoS defining characteristics. It is formally grounded on the π-Calculus for SoS [11].

As main contribution, this paper proposes a model-based refinement process to automatically generate architecture descriptions represented in SosADL from mKAOS mission models. The generated architecture descriptions encompass the whole structural view capable of achieving the mission.

The remainder of this paper is structured as follows. Section II provides an overview of both mKAOS and SosADL languages. Section III introduces our proposal for refining mission models towards architecture descriptions. Section IV briefly discusses related work. Section V contains some concluding remarks.

978-1-5090-5527-2/16 $31.00 © 2016 IEEE
DOI 10.1109/ICECCS.2016.40

II. BACKGROUND

A. Missions in SoS

mKAOS is a specialization of KAOS [6], a requirements specification language. The basic elements defined in KAOS are *goals*, which are related to requirements, conflicts, obstacles, and expectations in order to ensure that a requirement has at least one operational capability implementing it. mKAOS extends KAOS with constructs to represent mission-related concepts for supporting SoS mission modeling. As in KAOS, mKAOS separates models according to their concerns as well as allows overlaps to have cross-view perspectives. Besides specializing concepts defined in KAOS, mKAOS creates specific constructs suited to the SoS context, in particular emergent behaviors and missions.

Missions of SoS can be specified in mKAOS through six different models, each one with its own syntax and semantics.

The main mKAOS model is the *Mission Model*, which describes missions and expectations. The *Responsibility Model* concerns the description of both constituent systems, environment agents, and the assignment of missions/expectations to them. The *Object Model* specifies objects used by the SoS for data exchange and physical structures in terms of: (i) entities, which represent a data abstraction or physical entity; (ii) events that can be raised or handled; (iii) domain hypothesis, defined as constraints; and (iv) domain invariants, defined as constraints that must be held during SoS execution and further evolutions.

mKAOS also provides two *Capability Models*: the *Operational Capability Model* defines a set of operations that each constituent system is able to execute, i.e., their operational capabilities, whereas the *Communicational Capability Model* specifies the possible interactions and cooperation among constituent systems, the so-called communicational capabilities.

Finally, the *Emergent Behavior Model* describes emergent behaviors, specific features that are produced from the interaction between at least constituent systems. Table I summarizes the elements of the mKAOS models. More details about each of these models and their elements can be found in [17].

The *Mission Model* follows a tree structure in which leaf nodes represent individual missions and non-leaf nodes represent global missions, respectively assigned to constituent systems and to the SoS as a whole.

TABLE I. MKAOS MODELS AND RESPECTIVE ELEMENTS

mKAOS Model	**Model elements**
Mission Model	Mission, expectation
Responsibility Model	Constituent system, environment agent
Object Model	Entity, event, domain hypothesis, domain invariant
Operational Capability Model	Operational capability
Communication Capability Model	Communicational capability
Emergent Behavior Model	Emergent behavior

In this model, *expectations* represent assertions on the SoS environment that might influence the achievement of its missions.

In the Mission Model, *Refinement Links* establish a refinement relationship among missions, so that a given mission can be refined into other sub-missions and/or expectations. The assignment of missions to constituent systems is defined in a corresponding mKAOS *Responsibility Model*, in which each constituent system must have at least one assigned individual mission and each individual mission must be assigned to exactly one constituent system. In turn, expectations must be assigned to environment agents, which are external agents that somehow interfere on the SoS. Fine-grained mission-related specification can be expressed in mKAOS using the other provided models. mKAOS also allows defining relationships among missions.

B. Formally Describing SoS Software Architectures

When describing SoS software architectures, it is fundamental to consider: (i) both structural and behavioral definitions of its constituent systems and how they form together an SoS coalition; (ii) interactions among constituent systems; (iii) evolutions due to the dynamic scenarios in which an SoS operate; and (iv) properties, constraints, and quality attributes [1][13]. To cope with these concerns, SosADL [10][12] was defined as a formal language to comprehensively describe SoS software architectures while allowing for their automated, rigorous analysis. The formal foundations of SosADL rely on an extension of the π-calculus process algebra [11], thereby being a universal model of computation enhanced with SoS concerns.

One of the main characteristics of SoS software architectures is that the concrete constituent systems that are to be part of the SoS are partially known or even unknown at design time. For this reason, the constituent systems need to be bound dynamically. Thereby, an SoS software architecture is evolutionarily concretized only at runtime. To cope with this characteristic, SosADL allows describing SoS software architectures in an intentional, *abstract* way. This means that the architecture description expresses only the types of constituent systems required to accomplish the missions of the SoS as a whole (at design-time), but the concrete constituent systems themselves will be identified and evolutionarily incorporated into the SoS at runtime. Furthermore, the communication among constituent systems is said to be *mediated* in the sense that it is not solely restricted to communication (as in traditional systems), but it also allows for coordination, cooperation and collaboration.

SosADL uses a set of eleven elements, summarized in Table II: (i) *systems*; (ii) *gates*; (iii) *connections*; (iv) *assumptions*; (v) *guarantees*; (vi) *properties*; (vii) *behavior*; (viii) *mediators*; (ix) *duties*; (x) *coalitions*; and (xi) *bindings*. While coping with architectural concepts found in ADLs, the concepts defined in SosADL are aligned with the terminology of SoSs, fitting its domain semantics. More details about these elements can be found in [10] for the structural viewpoint and [12] for the behavioral viewpoint.

TABLE II. SUMMARY OF THE MAIN ELEMENTS OF THE SOSADL LANGUAGE

Element	Description
System	Abstract representation of a constituent system
Mediator	Abstract representation of the communication and coordination between two systems
Gate	Interaction point of a system with its environment
Duty	Obligation to be fulfilled by gates of systems that may interact with a mediator
Connection	Communication channel used by a system or mediator to send/receive data
Assumption	Property expected by a gate/duty to be satisfied by the environment
Guarantee	Property to be enforced by a system/mediator
Behavior	Functional capabilities of a system/mediator and how it interacts with the environment
Coalition	Representation of an SoS as an arrangment of systems and mediators forming its architecture
Binding	Dynamic attachment between a system and a mediator

The *system* concept is an abstract representation of a constituent system that may be part of the SoS, but that is not under its control due to its operational and managerial independences. A system encompasses gates (specified by properties in terms of assumptions and guarantees), and an internal behavior describing its operational capability to achieve its individual mission. A *gate* groups interaction points of a constituent system with its environment, encompassing at least one connection. A *connection* is a typed communication channel through which the constituent system sends or receives data. *Assumptions* express properties expected by a gate of a constituent system to be satisfied by the environment, e.g., rules related to provided/required data in gates. *Guarantees* describe properties that must be enforced by the constituent system, thereby being a way of representing specific properties at the architectural level. A *behavior* represents the operational capabilities of the system and how it interacts with the environment by sending/receiving data. The formally founded constructs for expressing behavior in SosADL [11] extends the ones defined in π-ADL [9], an ADL based on the π-calculus for formally describing dynamic software architectures of single systems under both structural and behavioral viewpoints.

In SosADL, a *mediator* is an architectural element under control of the SoS that mediates the communication, coordination, cooperation and collaboration among constituent systems, thus also promoting interoperability among them. Mediators differ from system-to-system connectors as they are used not only as mere communication channels, but also as elements responsible for the coordination, cooperation and collaboration among the interacting constituent systems. Therefore, mediators play an essential role in terms of making possible to an SoS to achieve its missions through emergent behaviors arising from such interactions. Similarly to systems, mediators can be also described abstractly, so that concrete mediators can be synthesized and deployed at runtime in order to cope with the highly dynamic environment of an SoS. A mediator definition encompasses a set of *duties*, which express obligations to be fulfilled by gates of constituent systems that may interact with the mediator. Moreover, a mediator allows defining properties, assumptions, guarantees, and an internal behavior.

A *coalition* represents the configuration of the SoS itself by intentionally defining how constituent systems and mediators can be temporarily arranged to compose the SoS. As constituent systems are not under the SoS control, it is necessary to specify how the mediators can be created and which systems will interact with them to define a concrete SoS. For this purpose, coalitions are composed by a set possible systems, mediators, and *bindings* that will be realized at runtime. A *binding* is the construct responsible for establishing dynamic connections between systems through mediators, in particular binding gates to duties. Such a dynamic nature of bindings is an important aspect for SoS since it is often not possible to foresee which concrete constituent systems will be connected through the mediators at runtime.

It is important to highlight that SosADL concerns focus on the architecture of the SoS as a whole. Therefore, the individual architectures of the constituent systems (even though desirable) are not mandatory in an SosADL description. This covers the fact that the internal architectures of the constituent systems are often unavailable, an typical case in the SoS domain. Nonetheless, the architecture of the SoS strongly depends on the interfaces of each constituent system, defined in terms of gates.

III. REFINING MISSION MODELS TO ARCHITECTURE DESCRIPTIONS

mKAOS was designed as a descriptive language for specifying missions of SoSs, focusing on what the system must be able to achieve instead of how it will achieve. Nevertheless, the descriptive elements of mKAOS refine mission definitions to the system level, assigning responsibilities and obligations of each constituent system. At this point, no further description related to how the system will achieve the existing missions is possible in mKAOS, indeed. Therefore, the SoS architectural description provides a new level of concretion by refining mKAOS models to an operational, architectural level. Although the proposed refinement relies on a mapping from missions to architecture, neither the mission model nor the architectural description provides sufficient information to represent the information from the each other, some data are not reflected in the architectural description during the refinement process (e.g., missions) and hence both models must be co-maintained for its own purposes.

Considering that mKAOS and SosADL provide different levels of abstractions for the SoS, the mapping process is based on the common concepts between both languages. These common concepts are the *interfaces* of the constituent systems and of the SoS. In mKAOS, an interface is a composition of the interfaces of the operational capabilities and communicational capabilities. On the other hand, SosADL defines interfaces as essential, explicit elements for defining the architecture of an SoS. It represents the interfaces as *gates* and associated *connections*, which are used for both structural and behavioral specifications.

The proposed mapping process is based on model-to-model (M2M) transformations [15][19], consisting in auto-

matically refining models to lower abstraction levels aiming to reflect solutions defined in higher levels. As the implementation of the mapping steps is intended to be automatic, all of them shall be programmatically executed using a M2M transformation. This ensures the traceability of the missions and simplify the architecture design process: the architect is concerned only on describing behavior and detailing further elements not related to the mission model. It is important to highlight that the transformation does not encompass all mKAOS elements neither the SosADL elements, but it still can be realized in both directions. Both mission and architectural models are complimentary to each other and they must be dependently maintained. In the proposed mapping process, we have chosen a constructive approach in which the refinement will produce a single architecture capable of achieving the required missions and making possible to emerge the desired behaviors. An alternative is to build a set of possible architectures and verify the conformance of each one with the mission model, but this approach is however computationally expensive.

The mapping process is divided into five steps, as depicted in Fig. 1.

Step 1 consists in identifying the data types used in the *Object Model* (entities and events) and defining them in SosADL. Step 2 involves identifying constituent systems from the *Responsibility Model* and defining them as possible constituent systems in SosADL. In Step 3, for each system, it is necessary to select the associated operational capabilities

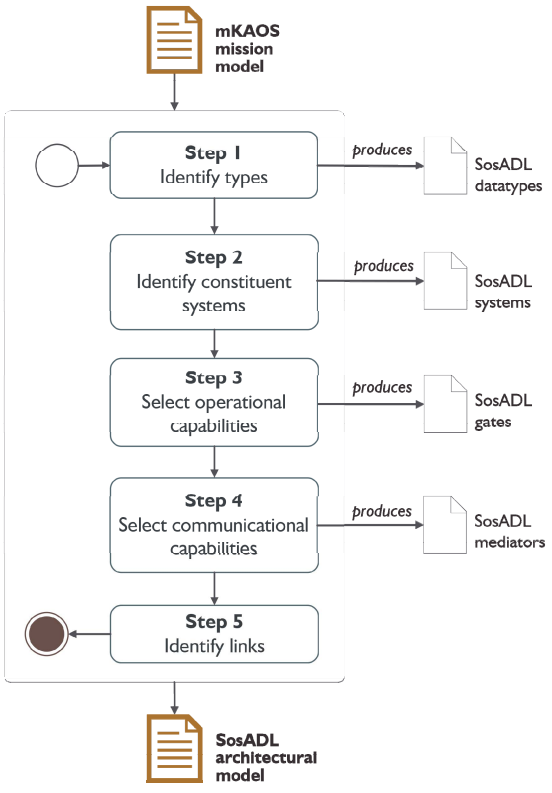

Fig. 1. Refinement process from mKAOS to SosADL models

specified in the *Operational Capability Model* and define a gate whose connections are defined for each input, output, and event. Input events will result in input connections while produced events will be mapped to output connections. In Step 4, each communicational capability defined in the *Communicational Capability Model* is concretized by a mediator whose duties are defined based on the input and outputs for the capability, similarly to the gate production. Inputs or outputs from communicational capabilities not used by any operational capability are described as inputs/outputs for the SoS as a whole. Finally, Step 5 consists in connecting constituent systems and mediators using the data association defined by input and output links in mKAOS, thereby establishing bindings in SosADL for each of these links. This last step involves the *Object Model* and both *Operational* and *Communicational Capability Models*, as well as the links between the objects and capabilities.

As an illustrative example, consider an SoS aimed to detect flash floods in a flood-prone area crossed by a river, with maximum confidence [10].

To achieve this global mission, such an SoS can combine information provided by multiple collaborating independent systems such as river monitoring systems and meteorological systems.

Within the SoS, river monitoring systems composed of sensor networks monitor the river water level as an indicator of flood while meteorological systems comprise weather stations and satellites to collect and analyze atmospheric parameters for weather conditions.

As depicted in Fig. 3 (top), these missions and systems responsible for achieving them can be modeled using overlapped *Mission* and *Responsibility Models* of mKAOS. A river monitoring system is capable of providing information about the state of the river and a meteorological system is capable of producing weather bulletins. These data can be aggregated to enable the SoS to provide more precise information about the possible risk of flood.

These operational and communicational capabilities can be represented in a *Capability Model* of mKAOS, as shown in Fig. 3.

Mapping these mKAOS models by means of the proposed process will result in an architectural model in SosADL composed of constituent systems, mediators, their respective gates and duties, and bindings attaching them.

Fig. 3 illustrates what would be the result of refining operational and communicational capabilities defined in a mKAOS *Capability Model* towards an architectural model in SosADL: (i) the *River Monitoring System* and the *Meteorological System* are mapped to *systems*; (ii) the respective *ProvideRiverInformation* and *ProduceWeatherBulletin* operational capabilities are mapped to *gates* interfacing to systems and thereby to their operational capabilities; (iii) the communicational capability that enables these systems to interact with each other (*AggregateData*) is mapped to a *mediator*; and (iv) *bindings* are created to attach these elements in order to form a *coalition* representing the SoS as a whole.

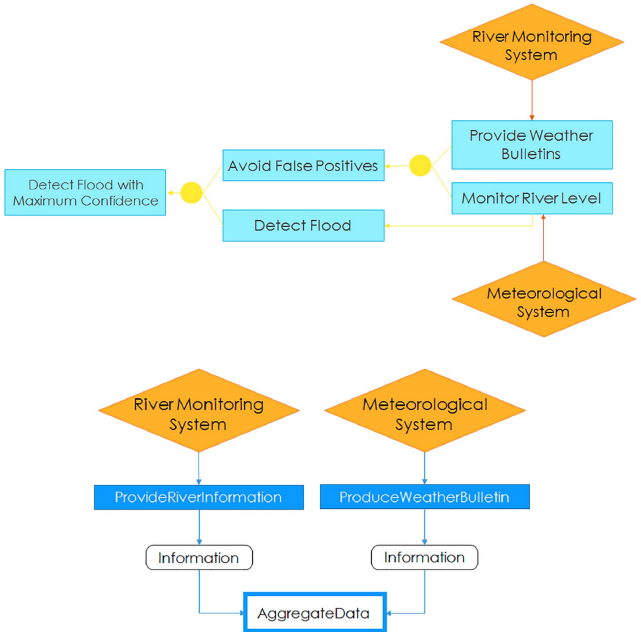

Fig. 2. Overlapped *Mission Model* and *Responsibility Model* (top) and *Capability Model* (bottom) for the flood monitoring SoS.

Table III summarizes the correspondences between the mKAOS and SosADL elements, implemented by the mapping process. As mKAOS assumes that emergent behaviors arise from the communicational capabilities and the mapping process promotes an implementation for each communicational capability as mediators, it is expected for the architecture to emerge the desired behaviors. In turn, individual missions are direct consequence of operational capabilities of constituent systems and hence they are covered by the mapping of these capabilities to the gates of the constituent systems that compose the architecture.

The generated SoS architecture in SosADL, produced from the mapping process, needs to have its behavior defined since mKAOS does not cover behavioral concerns of SoS. This step is essentially manual because it largely depends on architectural decisions regarding how the constituent systems will implement their capabilities. It is also fundamental for the architect to consider non-functional requirements (NFRs), identify which constituent systems and mediators are involved in each requirement, and define and apply assumptions and guarantees for them, as well as for

TABLE III. CORRESPONDENCES BETWEEN THE ELEMENTS OF THE MKAOS AND SOSADL LANGUAGES

mKAOS	**SosADL**
Constituent system	*System*
Communicational capability	*Mediator*
Operational capability	*Gate (in system)*
Input/output/event	*Input/output connection*
Entity	*Data type*
Event	*Data type*

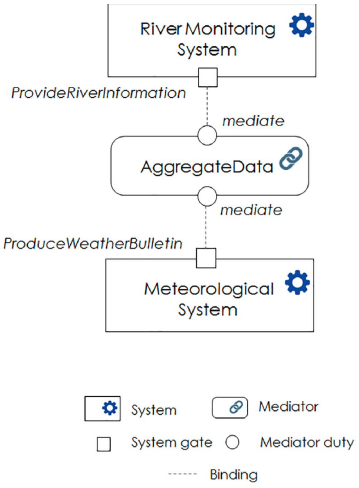

Fig. 3. Result of the refinement of the mKAOS *Capability Model* (see Fig. 2) to a partial architectural model in SosADL.

the SoS architecture itself. These NFRs can be manually derived from mKAOS domain hypothesis and invariants.

IV. RELATED WORK

This work is pioneer with respect to the refinement of mission models towards generating architectural models in SoS. To the best of our knowledge, the only relevant work that is worth highlighting is COMPASS (*Comprehensive Modeling for Advanced Systems of Systems*) [5], a manual methodology to produce and evaluate SoS architectural models from requirements. Such a methodology consists in using different tools to produce and analyze different models expressed in the OMG's Systems Modeling Language (SysML) complemented by a model in the COMPASS Modeling Language (CML), a formal specification language specifically designed to support SoS modeling and analysis. SysML is used to model the constituent systems and interfaces among them in an SoS whereas CML is used to enrich these specifications with interaction contracts. Indeed, the authors of CML have acknowledged that it is a low-level formal language and mapping SysML models to CML produce less intelligible descriptions.

In terms of refinement process, our proposal differs from the one adopted in COMPASS regarding several aspects. First, we use the mission concept, which is more suitable for use in the SoS context [18]. Second, the SysML language is not tailored to SoS as it lacks some important features required for SoS, such as emergent behaviors. Third, the solution provided by COMPASS is more concerned with concrete architectures for SoS, whereas our approach produces abstract architectures to be further concretized according to the availability of the constituent systems forming the SoS. Oppositely to COMPASS, our approach does not require knowing the actual constituent systems a priori (as it is often expected for SoSs). The COMPASS methodology also needs additional information about constituent systems, such as stakeholders and specificities regarding communication capabilities.

V. Conclusion

In this paper, we have introduced a model-based refinement methodology to produce architectural descriptions from mission models described using mKAOS, a pioneer descriptive language for SoS mission definition. The proposed approach relies on a mapping process that results in architectural descriptions expressed in SosADL, a formal ADL for the specification of SoS. Focusing on capabilities, our refinement process derives a structural description from both operational capabilities of the constituent systems and communicational capabilities of the SoS. Similarly to the existing approaches for deriving software architectures from requirements, the proposed mapping process relies on a top-down approach that allows producing SoS software architectures based on a high-level description of the constituent systems. The mapping process also ensures traceability between the mission and architectural models as it is based on a model transformation, thereby enabling architects to precisely identify which pieces of the software architecture are responsible for realizing each mission.

As future work, the model transformation rules from mKAOS to SosADL will continued to be integrated in a toolkit for model-based development of SoSs. We also intend to implement a verification and conformance mechanism to support the co-evolution of SoS missions and SoS architectures.

It is worth mentioning that mKAOS is a semi-formal notation, whose formalization is planned in future work. We will investigate the use of a temporal logic of evolving systems [14] as the basis for this formalization.

Also planned as future work is the application of mKAOS together with SosADL in several industrial-scale projects. They include joint work with DCNS for applying SosADL to architect naval SoSs, with IBM for applying SosADL to architect smart-farms in cooperative settings, and with SEGULA for applying SosADL to architect SoSs in the transport domain.

Acknowledgment

This work was partially supported by the Brazilian National Agency of Petroleum, Natural Gas and Biofuels (PRH-22/ANP/MCTI Program) and the Brazilian National Council for Scientific and Technological Development (CNPq) under grant 308725/2013-1.

References

[1] Batista, T.: “Challenges for SoS Architecture Description”, Proc. *1st ACM International Workshop on Software Engineering for Systems-of-Systems (SESoS)*, F. Oquendo et al. (eds.), Montpellier, France, July 2013, pp. 35-37.

[2] Boehm, B.; Lane, J. A.: “21st Century Processes for Acquiring 21st Century Software-intensive Systems-of-Systems”, *CrossTalk: The Journal of Defense Software Engineering*, Vol. 19 (5), May 2006, pp. 4-9.

[3] Cavalcante, E.; Batista, T.V.; Oquendo, F.: “Supporting Dynamic Software Architectures: From Architectural Description to Implementation”, *Proc. of the 12th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, Montreal, Canada, May 2015 pp. 31-40.

[4] Cavalcante, E.; Quilbeuf, J.; Traonouez, L.M.; Oquendo, F.; Batista, T.V.; Legay, A.: “Statistical Model Checking of Dynamic Software Architectures”, *Proc. of the 10th European Conference on Software Architecture (ECSA)*, LNCS, Springer, Copenhagen, Denmark, November 2016.

[5] COMPASS: Comprehensive Modelling for Advanced Systems of Systems, http://www.compass-research.eu

[6] Darimont, R.; van Lamsweerde, A.: “Formal Refinement Patterns for Goal-driven Requirements Elaboration”, *Proc. 4th ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE)*, San Francisco, CA, USA, October 1996, pp. 179-190.

[7] Guessi, M.; Nakagawa, E.Y.; Oquendo, F.: “A Systematic Literature Review on the Description of Software Architec-tures for Systems-of-Systems”, *Proc. of the 30th ACM Symposium on Applied Computing (SAC)*, Salamanca, Spain, April 2015, pp. 1-8.

[8] Maier, M.: “Architecting Principles for Systems-of-Systems”, *Systems Engineering Journal*, Vol. 1 (4), February 1998, pp. 267-284.

[9] Oquendo, F.: “π-ADL: Architecture Description Language based on the Higher-order Typed π-Calculus for Specifying Dynamic and Mobile Software Architectures”, *ACM SIGSOFT Software Engineering Notes*, Vol. 29 (3), May 2004, pp. 1-14.

[10] Oquendo, F.: “Formally Describing the Software Architecture of Systems-of-Systems with SosADL”, *Proc. of the 11th IEEE System-of-Systems Engineering Conference (SoSE)*, Kongsberg, Norway, June 2016, pp. 1-6.

[11] Oquendo, F.: “π-Calculus for SoS: A Foundation for Formally Describing Software-intensive Systems-of-Systems”, *Proc. of the 11th IEEE System-of-Systems Engineering Conference (SoSE)*, Kongsberg, Norway, June 2016, pp. 7-12.

[12] Oquendo, F.: “Formally Describing the Architectural Behavior of Software-intensive Systems-of-Systems with SosADL”, *Proc. 21th IEEE International Conference on Engineering of Complex Computer Systems*, Dubai, UAE, November 2016.

[13] Oquendo, F.: “Software Architecture Challenges and Emerging Research in Software-intensive Systems-of-Systems”, *Proc. of the 10th European Conference on Software Architecture (ECSA)*, LNCS, Springer, Copenhagen, Denmark, November 2016.

[14] Quilbeuf, J.; Cavalcante, E.; Traonouez, L.M.; Oquendo, F.; Batista, T.V.; Legay, A.: “A Logic for Statistical Model Checking of Dynamic Software Architectures”, *Proc. of the 7th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISOLA)*, Springer, Corfu, Greece, October 2016.

[15] Sendall, S.; Kozaczynski, W.: “Model Transformation: The Heart and Soul of Model-Driven Software Development”, IEEE Software, Vol. 20 (5), September 2003, pp. 42-45.

[16] Silva, E.; Batista, T.; Cavalcante, E.: “A Mission-oriented Tool for System-of-Systems Modeling”, *Proc. 3rd ACM/IEEE International Workshop on Software Engineering for Systems-of-Systems (SESoS),* Florence, Italy, May 2015, pp. 31-36.

[17] Silva, E.; Batista, T.; Oquendo, F.: “A Mission-oriented Approach for Designing System-of-Systems”, *Proc. 10th IEEE System-of-Systems Engineering Conference (SoSE)*, San Antonio, TX, USA, May 2015, pp. 346-351.

[18] Silva, E.; Cavalcante, E.; Batista, T.; Oquendo, F.; Delicato, F. C.; Pires, P. F.: “On the Characterization of Missions of Systems-of-Systems”, *Proc. 2014 European Conference on Software Architecture (ECSA) Workshops*, ACM, Vienna, Austria, August 2014, pp. 1-8.

[19] Völter, M.; Stahl, T.; Bettin, J.; Haase, A.; Helsen, S.: *Model-Driven Software Development: Technology, Engineering, Management*, John Wiley & Sons, 2006.