MBA: A Framework for Building Systems of Systems

Gregory Moro Puppi Wanderley, Marie-Hélène Abel, Emerson Cabrera Paraiso, Jean-Paul Barthès

MBA: A Framework for Building Systems of Systems

Gregory Moro Puppi Wanderley*, Marie-Hélène Abel*, Emerson Cabrera Paraiso†, Jean-Paul A. Barthès*
*Sorbonne Universités, Université de Technologie de Compiègne,
CNRS, UMR 7253 Heudiasyc, Compiègne, France
Email: {gregory.wanderley, marie-helene.abel, barthes}@utc.fr
†Pontifícia Universidade Católica do Paraná
PPGIa - Graduate Program in Informatics, Curitiba, Brazil
Email: paraiso@ppgia.pucpr.br

***Abstract*—Although the concept of System of Systems (SoS) has become quite popular, most applications are still hand crafted. In this paper we present a framework, called MBA for Memory-Broker-Agent, addressing the development of systems of systems from an engineering perspective. The main features of the framework result from the experience gained from building an SoS for developing software collaboratively. In the paper we recall the requirements for building an SoS and show how they can be met by using a multi-agent substrate. The MBA framework is a first step towards proposing a generic platform for developing systems of systems.**

***Index Terms*—Systems of Systems, Architecture, Multi-agent system, Knowledge management, User interface**

Since the first introduction of the General System Theory (GST) by Ludwig von Bertalanffy in the 40s [1] a number of complex systems have been developed in various domains. Also, the need of building systems successfully has led to the emergence of the field of systems engineering and the creation of the International Council on Systems Engineering (INCOSE).

Today complex applications require to let several systems that have been developed independently work together, leading to the concept of system of systems (SoS). The notion of SoS from a systems engineering perspective, focused on connecting independent systems together, gained momentum with the United States Strategic Defense Initiative (SDI) from the late 1980s [2].

Nowadays, SoS has been applied and developed in a number of different domains, e.g., to improve the quality of the code [3], to assess the sustainability of different sources of energy [4], to support manufacturing [5], to improve understanding medication prescriptions [6], to support tourism [7], to support military activities ([8]; [9]; [10]), or to help developing particle accelerator facilities [11].

However, most SoS are still hand crafted and no consensus has yet been reached about a precise definition of what they are ([12]; [13]; [14]; [15]; [16]). In this paper, we present a framework to facilitate the development of systems of systems from an engineering perspective. We determine essential elements and discuss how the corresponding architecture answers most of the requirements for building systems of systems.

I. Requirements for building an SoS

The lack of consensus in the definition of systems of systems has lead researchers to start basing their definitions on SoS characteristics, which according to Maier [17] are the following.

- *Operational Independence*: all the constituent systems of an SoS can often deliver their functionalities when not working with other constituents.
- *Managerial Independence*: each constituent system of an SoS is governed by its own rules rather than by others external to the constituent.
- *Evolutionary Development*: functions and purposes of an SoS can dynamically change and constituent systems can be added or removed to fit them.
- *Emergent Behaviour*: an SoS is capable of delivering new functionalities that are the result of the constituent systems working together.
- *Geographic Distribution*: constituent systems of an SoS are geographically distributed, meaning that they can readily exchange only information and not substantial quantities of mass or energy.

Note that such features are also cited by the INCOSE [18] as useful to define what an SoS can be.

Because of such characteristics, developing an SoS meets several challenges like interoperability, robustness, knowledge management, user involvement, or evolution.

a) Interoperability: In an SoS, constituent systems need to exchange information and cooperate. Interoperability can be defined as the ability of distinct systems to share semantically compatible information and then process and manage such information in semantically compatible ways, enabling users to perform desired tasks (Zeigler et al. [19]; Madni and Sievers [20]). Providing interoperability between constituent systems that do not interoperate natively can require substantial effort and cost. To support SoS interoperability, communication protocols need to be used, simplifying and managing the connections among heterogeneous systems.

b) Robustness: Because of the evolutionary nature of SoS, interdependencies and relations between SoS constituents can be modified, meaning that changes can occur in an unpre-

dictable manner (De Laurentis [21]). Thus, it is paramount to consider robustness in SoS. In the context of SoS, robustness can be defined as the ability to deliver capability in unknown future conditions.

c) Knowledge management: Performing Knowledge Management (KM) is essential as SoS change over time. Because systems and users may change during SoS life cycle, crucial knowledge must be collected, organized and preserved.

d) User dimension: Besides being an important characteristic, the user dimension is also a challenge in SoS. Unlike in monolithic systems in which users have predefined interfaces, in an SoS users interact through changing interfaces with other systems or with users of the SoS (Madni and Sievers [20]). Such a dynamic behavior occurs mainly because systems can be added, removed or replaced in an SoS. Moreover, interactions can occur through interfaces in external systems connected to the SoS such as a Web server or another application. Furthermore, information for decision-making should be given on behalf of individuals, i.e., should take a user perspective, providing customized information perhaps through AI-based systems.

e) Evolution: Evolution means changing an SoS, for instance, by adding, removing, replacing or modifying its constituent systems. According to Agarwal et al. [22], some of the needs for evolving an SoS can be for: (i) improving SoS performance; (ii) rendering constituent systems interoperable; (iii) including additional requirements for the SoS goal; (iv) handling evolution in the constituent systems; and (v) adding new functionalities to the constituent systems. A usual practice recommended to handle the evolution in SoS is to leave constituent systems loosely coupled.

The following section proposes a new approach, bringing answers to the challenges we just mentioned.

II. The MBA Architecture

Considering the characteristics and challenges for building systems of systems, one cannot but think of multi-agents systems (MAS). Indeed, although an agent in an MAS usually cannot work meaningfully without the support of the other agents, many MAS features constitute good candidates for supporting an SoS architecture. The approach thus consists of building systems of systems on top of a multi-agent layer that will provide the needed mechanisms for answering the requirements. Such an approach is inspired by the PACT (Palo Alto Collaborative Testbed) project (Cutkosky et al. [23]) that introduced the idea of facilitator and allowed Gruber to develop the concept of ontology.

A. Overall Approach

The main idea is to render the component systems interoperable by using facilitator agents, one for each system. Then, we link the agents to brokers (Park et al. [24]) that will take care of organizing exchanges using a standard protocol. The MAS platform will also offer the possibility of adding Personal Assistant agents to interface with users, and agents in charge of recording information necessary for managing knowledge. Thus, the interoperability and robustness issues will be addressed by agentifying the component systems using facilitators, brokers, a standard protocol and ontologies; knowledge management, by the existence of agents in charge of memory; user dimension, by personal assistant agents; and evolution, by a possibility of plug and play of the component systems. Hence, the name of the proposed framework: MBA for Memory-Broker-Agents.

B. The MBA Framework

The MBA model can be defined as a tuple $\psi := (M, B, A, Y)$, where

- M, B, A are finite sets of memories, brokers, and agents, respectively.
- Y is a ternary relation among them, i.e., $Y \subseteq M \times B \times A$. This relation specifies the coupling among parts.

The MBA architecture is a domain-independent core architecture, meaning that it is extended according to the domain, goals and systems of the SoS being developed. In the architecture, three main elements are distinguished: "system," "broker" and "memory." A minimal example of the proposed architecture is shown in Fig. 1.

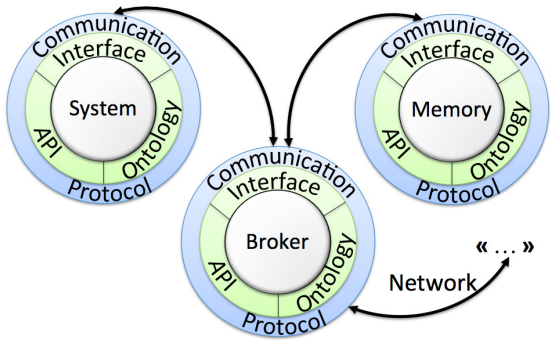

Fig. 1. The MBA architecture proposed in this research.

The *system* element represents a component system of an SoS. To become a constituent system of an MBA SoS, a system must provide at least an API (Application Programming Interface). After being interfaced through a facilitator agent, the system will be augmented by a communication protocol, ontologies, and optional interfaces.

The communication protocol is used to exchange messages and information between the system and the other constituent systems in the MBA SoS.

The *broker* element is intended to receive requests from component systems, try to find potential providers, and then once the tasks are allocated, transfer the results back to the callers. In addition to providing loose coupling between systems, an important characteristic of the broker is that it can be a single agent or a set of broker agents. Using several brokers can avoid overloading a single broker, thus enhancing efficiency, and making the MBA architecture more robust. In addition brokers use a Contract-Net protocol (Smith [25]), which allows including component systems offering

competing services, selecting from the most efficient one, or easily phasing out a subsystem when a new version appears.

The *memory* element is there to capitalize and manage knowledge concerning the SoS and its domain. In our approach, it is possible to have as many memories as needed or required by a given domain, including redundant elements. The main advantage of memories in our approach is that we are able to reuse knowledge from the SoS and its domain. Memories are usually external systems. Currently, the MBA architecture provides facilitator agents ready to interface different kinds of memories based on object, relational, key-value and triplestore data models. The advantage of doing this, is that our approach facilitates the work of SoS architects, letting them customize such facilitator agents for their application.

Once all elements have been "agentified," one must provide a standard protocol for requesting services and receiving answers and ensure both syntactic and semantic interoperability.

C. The MBA Communication Protocol

The low level protocol is provided by the supporting multi-agent platform. At a higher level, requests for services are modelled using *work orders* and answer formats by *answer patterns*.

1) Work Orders: A *work order* describes a request sent to a broker. Formally, it can be defined as follows:

```
<work order> ::= (<message-content>[, <action>],
                  <timeout>[, <C-net strategy>])
```

where,

- `<message-content>` is the content or arguments of the work order detailed below.
- `<action>` is an optional parameter, being the name (a keyword) of a requested functionality. To succeed in requesting `<action>`, the requester must appear somewhere in the ontology of the potential providers.
- `<timeout>` is a number representing the maximum amount of time a requester will wait to receive an answer to its request.
- `<C-net strategy>` is a parameter used by requesters to specify the strategy used in a Contract Net for getting answers.

Because there may be different reasons for a request not to be answered, e.g. the facilitator agents interfacing the systems are fully busy working on several requests, or they simply do not want to answer, the MBA architecture adopts the approach that systems are permitted not to answer requests. Consequently, it is necessary to allow specifying timeouts in the work order.

The arguments or content of a work order, i.e., `<message-content>`, can be defined as follows:

```
<message-content> ::= ([<query>], [<data>],
                       [<pattern>], <language>)
```

where,

- `<query>` represents a request made in a query format (the ontologies of the facilitator agents of requesters and providers must be at least compatible).
- `<data>` stands for some input data.
- `<pattern>` an answer pattern determining how results should be structured.
- `<language>` the language used in the the message content.

2) Answer Patterns: An *answer pattern* in the MBA architecture specifies how the results from a request should be structured in order to be understood by the requester. It is a list of linguistic cues expressed as a tree of ontological terms. These terms are concepts and properties taken from the requester's ontology. The pattern gives semantics to requests and results.

```
<pattern> ::= (<concept> ({<attribute>}*))
                          | {<relation> <pattern>}*
```

D. Ontologies

Ontologies constitute the backbone of our architecture. They play four different roles: (i) modeling the SoS domain; (ii) modeling the users of constituent systems; (iii) decoding interactions in natural language; (iv) handling the semantics of the communication protocol.

Because an SoS is composed of systems that were built independently and because its structure will change over time, it is not possible to construct a single ontology that could homogenize all its different parts. It is possible however to align parts of the different ontologies within the application to ensure a minimal understanding.

A second role of the ontologies is to model users of the component systems. Representing the behavior of the user interacting with a component system will help provide personalized support for improving her interaction with the SoS. This can be done in connection with the user's personal assistant.

Ontologies are also essential for decoding interactions in natural language. On the system side, the goal in this case is to help decoding, interpreting and understanding the utterances between users and their PAs.

Ontologies are also necessary to interpret expressions of the message content language, ensuring semantic interoperability.

The ontologies used to support the semantics are kept within the facilitator agents. It is important to highlight that such ontologies can differ in each agent, being specialized according to the characteristics of the systems they are interfacing. Thus, the ontologies of two given facilitator agents need not be the same but must have a minimal degree of compatibility to allow exchanging information coherently.

E. Summarizing Process

After accomplishing the allocated tasks, providers are responsible for structuring the results in a format understandable by the requester. To do that, they must perform a *summarizing process*, which consists of filling an answer pattern with the

relevant information. Each provider, through its facilitator, must align its ontology with the content of the answer pattern in the work order, then organize its data to structure it according to the pattern.

F. User Dimension

SoS have a number of human users who have two kinds of interfaces: an interface with a component system, and an interface with the SoS. Because point and click interfaces are not efficient in complex environments, and because with the MAS substrate we can use personal assistant agents (PA), a natural way of interacting with the SoS is by using natural language written or spoken.

A PA is an important asset since it can act proactively, provide customized support, reduce the user's cognitive load, help to increase cooperation with other SoS users, and handle multimodal interactions. Moreover it has been shown that natural language dialogues in a professional context are not very difficult to implement (Barthès [26], Fuckner et al. [27]). Finally, speech-to-text and text-to-speech programs have made enough progress to be now integrated in such kinds of interfaces (Jones et al. [28]).

III. Case Study

We built a prototype in the domain of software development to assess the different problems that could arise.

A. Overview

The objective of software development is to support team members working collaboratively for improving the quality of the code they produce in a given project. The purpose of the targeted SoS was the following: While developers are writing their code, knowledge about the code quality is capitalized automatically and in a non intrusive way; then, feedback is provided showing possible quality issues and recommendations for addressing them; in parallel, managers receive information about the quality of projects or the quality of the code through charts or tables, to help them make decisions.

The SoS shown in Fig. 2 consists of coding systems, a code analysis system, a versioning system, different kinds of DataBases (DB) such as relational, object, key-value, and triplestore, MBA Browsers systems, and a Web search system. All the systems are operational and managerial independent, but cooperate together aiming to improve the quality of the code.

The overall SoS works as follows. Developers use their IDE to produce (Java) code. They can write code of a given project that has its quality considered by the SoS, or to develop code in parallel related to other projects which are not taken into account by the SoS. The produced code is kept in a versioning system which can store code in real-time from a number of different projects. The analysis system retrieves the code from the versioning system, performing quality analysis. At the same time the analysis system cooperates with the SoS, it can also provide analysis to code from different teams and projects.

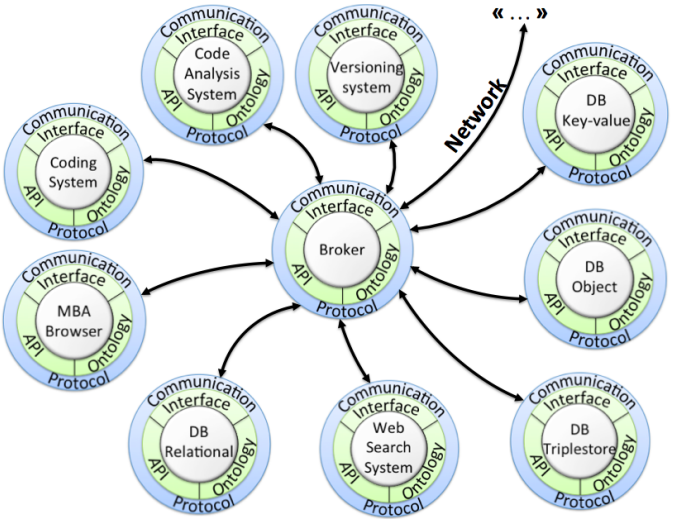

Fig. 2. The proposed MBA architecture applied to the context of collaborative software development.

When weaknesses are found, recommendations are extracted from the various databases and from the Web which may also work in parallel for storing or retrieving information outside the SoS. After that, the recommendations are forwarded to the developer through her coding system. An interactive vocal interface allows communicating with the system for requesting services or with other developers to obtain information. The project manager has an external system with a Web interface handled by a Personal Assistant, allowing her to request and receive information from the SoS, for instance, to view the progress of the developments and the problems encountered.

B. Implementation

The coding system used by each developer was Eclipse[1]. The versioning system was github[2]. The analysis system was a multi-agent system applying rules to compute the quality of the produces Java code. The relational database was MySQL[3], the object database was AllegroCache[4], the key-value database was Redis[5], the triplestore was actually a knowledge management system handling documents viewed as resources MEMORAe[6] (Abel [29]). The supporting MAS platform was OMAS[7] (Barthès [30]). Facilitators were instances of the OMAS Transfer Agents, interfaces were implemented as instances of OMAS Personal Assistant Agents. The Broker was an instance of OMAS Service Agent. All Web interfaces were used the possibility of viewing the MAS platform as a Web server. Interactions used the platform Contract Net facility. Vocal interface was implemented using speech-to-text and text-to-speech software.

[1]https://www.eclipse.org/
[2]https://github.com/
[3]https://www.mysql.com/
[4]https://franz.com/products/allegrocache/
[5]https://redis.io/
[6]http://memorae.hds.utc.fr/demo/labo/
[7]http://www.utc.fr/∼barthes/OMAS/

C. Discussion

Building the prototype of SoS for developing software collaboratively allowed us to determine the important feature a generic platform should offer. The role of the MAS during implementation was crucial since it provided agent models for building the facilitators, the personal assistant agents for interfacing humans and the Contract-Net protocol for exchanging messages. The OMAS environment also provided support for developing ontologies easily and a reasoning process similar to Jena or SPARQL. Work orders were built in a trivial fashion, and OMAS offered the summarizing mechanism to let facilitators format their answers. Natural language processing and dialogue mechanism were also supported by existing structures. A difficult point however was to build the actual personal assistant dialogues, which required a long and tedious work. Building facilitators used the available Transfer Agent mechanism, but required to install a specific ontology and translators for matching the inter-agent message structure to the idiosyncrasies of each element system, for example, matching the ontology with relational tables for the MySQL facilitator.

An important discovery was that we could replace the broker agent by using conditional addressing allowing to deliver messages only to agents that satisfy certain conditions formulated using the sender ontology and the ontology query structure. The OMAS platform is quite efficient with broadcast messages or Contract-Net messages since the call for bids can be done with a single message, as well as granting a task. This approach, defining a virtual broker, preserves the P2P nature of the exchanges among agents.

IV. Global Discussion

After presenting the approach, let us see now how it answers some of the challenges listed at the beginning of the paper.

- *Interoperability* is provided by the underlying multi-agent platform and the use of facilitators to insert external systems or legacy systems. The MAS platform also allows geographical distribution.
- *Robustness* is improved by the use of Contract-Net, which allows inserting new components in parallel to existing ones as well as optimizing services.
- *Knowledge management* must be implemented in the framework of each application but can rely on the interface of databases (relational, object, or other) or more sophisticated systems like MEMORAe.
- *User dimension* is taken into account by the possibility of interfacing users to the SoS through Personal Assistant agents, conducting exchanges using natural language in a textual or vocal mode. Personal assistants can be proactive and make use of user profiles to better customize interactions.
- *Evolution* is favored by the loose coupling of element systems and the use of standard Work Orders and Answer Patterns in the protocol. Loose coupling ensures the operational and managerial independence of the component systems.

Note that the proposed framework does not apply to all the characteristics of a system of systems, for example concerning emergence. Such characteristics relate to applications. However, by trying to answer challenges, we hope that the MBA framework will lead to develop systems of systems displaying the characteristics mentioned at the beginning of the paper.

Concerning related work, there is not enough room here to mention the numerous papers that have discussed the architecture of an SoS. We examine some recent ones.

Many approaches rely on centralized components, for instance, to store and provide the services or functionalities used by systems in the SoS or to render constituent systems interoperable focusing on semantics and translations (e.g. Perez et al. [31] or Varga et al. [32]). The use of a centralized component makes the architecture more brittle, creating single points of failure, which is even more risky in the case of SoS because of emergent and unpredictable behavior. Furthermore, the central component can be overloaded with requests, thus the SoS can suffer from bottlenecks, which can affect the expected results. In addition, the use of central elements leads to tight coupling, which is not appropriate for coping with the evolution of the system.

Regarding interoperability, most approaches use or recommend the use of standards. Some works like (Wong et al. [33]) use several protocols or standards at the same time. In this case, message translation is done within central components. This increases the complexity, effort and cost, and each time a system using a different protocol is connected to the SoS, new translation mechanisms need to be created and added to the central element. In some cases, in order to implement the changes, the central element needs to be stopped, bringing the entire SoS to a halt.

Some works take a more conceptual view regarding architectural support for SoS. Some authors propose languages to formally describe the architecture (Oquendo [34]), or techniques to optimize and provide decision-making using SoS architectures (Agarwal et al. [22]). Others try to study architectural patterns (Ingram et al. [35]). Others still, like Ge et al. [36] propose using well-known architecture frameworks such as DoDAF. However, such frameworks represent the architecture statically and focus too much on what should be described rather than on the practical problems. They focus mainly on a conceptual level, i.e., describing and documenting the architectures, rather than a practical point of view for supporting the development of SoS.

Regarding user dimension, most approaches use the WIMP paradigm (Windows, icons, menus, pointer), difficult to use in the case of SoS as information is exchanged among different kinds of systems. Moreover, the use of rigid interfaces cannot be appropriate for SoS as they change over time. Some authors offer interaction in natural language, but on a limited scale.

Although robustness is an important issue, we did not find an approach that tries to support SoS robustness from a practical point of view, i.e., directly during SoS operation. For instance, none of the approaches try to bring back an SoS to a coherent state after its constituent systems have gone down.

This can be important since once systems are reconnected to the SoS their state may no longer be consistent with the rest of the SoS, and thus their information can be incoherent and spread over all the SoS.

Over the years, a growing number of authors have been trying to categorize SoS, for instance, to guide the selection of architecting principles. Four main categories (Maier [17]; Dahmann and Baldwin [37]) based on the authority relationships between the SoS and the constituent systems have been widely adopted (Dahmann and Roedler [38]):

- *Directed*: In these SoS, the constituent systems are subordinated to a central authority to fulfill a specific purpose. The constituent systems of the SoS have the ability to operate independently, but are managed to satisfy the target purpose.
- *Collaborative*: The constituent systems interact and collaborate voluntarily to fulfill the agreed common purpose. In this category of SoS, a central management organization does not have coercive power to run the system.
- *Acknowledged*: It is a hybrid of the directed and collaborative SoS. There is a management authority at both the SoS and the system levels. The acknowledged SoS have clear purposes, management and resources. The constituent systems continue as independent entities, pursuing their own goals with independent management, resources, stakeholders. There is a concurrent management. Competing interests and priorities may arise.
- *Virtual*: In this type of SoS, there is a lack of both central management authority and centrally agreed upon purposes. Large-scale behavior emerges, and may be desirable, but the SoS must rely upon relatively invisible mechanisms to maintain it. A virtual SoS may be deliberate or accidental.

Now, how could the MBA framework help to build such types of SoS?

First, it is important to note that the constituent systems of an SoS built through our framework are not naturally constrained by any form of managerial control. The main reason is because our goal is to provide a generic approach, thus we avoid imposing such a constraint. However, we describe now how our framework could be used for building SoS of the four types mentioned above.

Directed SoS: when building a directed SoS with the MBA framework, the central authority could use external systems with Personal Assistant (PA) agents to control the SoS by requesting or receiving its functionalities. Our approach already provides such an external system which is the MBA Browser used by software managers in the case study of Section III. The access to the SoS functionalities by both, the central authority and constituent systems, could be customized in facilitators of constituents and dialogs of the PAs.

Collaborative SoS: for building a collaborative SoS with our approach, an architect may use the MBA framework directly. For instance, she needs to interface potential constituent systems with facilitators and link them with brokers, as described in this paper. The reason for that is because inherently an SoS built through the MBA framework is not compelled to central authorities. An example of collaborative system built using our approach can the case study of collaborative software development described in Section III.

Acknowledged SoS: in an acknowledged SoS, the central authority could also use external systems interacting through PAs, as we have recommended for the Directed SoS type. When dealing with competing interests and priorities, constituent systems could rely on a more technical level on the Work Order protocol, by using timeouts and parametrizing facilitators to answer or not requests. However, if a more sophisticated approach is needed, then it must be implemented by the architect, for instance, specifying in the facilitator agents how to handle the incoming Work Orders.

Virtual SoS: for building a virtual SoS with the MBA framework, first the architect would be required to interface potential constituent systems with facilitators, as usual. Then, because a virtual SoS does not have a central authority and not even a specific purpose, the facilitators could be customized for allowing the constituents request and receive perhaps all functionalities from each other freely.

Following this line of categorizing SoS through types, the MBA SoS we built in the case study of Section III could be considered of the collaborative type. The main reasons are that, first the constituent systems do not rely on central authorities controlling the SoS. That is, the interactions between them are performed mainly according to the SoS goal of improving code quality. The stakeholders such as software development managers and developers are free to request the SoS functionalities, in consonance with their roles in the SoS domain. Moreover, in the application level, managers and developers can collaborate, for instance, by suggesting recommendations to improve the code quality. However, developers are free to accept or not such suggestions.

The main differences between the works of the literature to the approach proposed by this research is that we propose a domain-independent peer-to-peer (P2P) architecture with loose coupling between its elements, focusing on the development of an SoS from a practical point of view. Our approach does not rely on centralized components or keep references between each its elements. Moreover, thanks to the loose coupling provided by our architecture, the SoS developed with it can be easily adapted to different domains. Furthermore, our approach provides basic elements for knowledge capitalization and management, and uses a single communication protocol providing syntactic and semantic interoperability between constituent systems. It also takes into account the user dimension by providing proactive interfaces capable of interacting with SoS users through dialogues with Personal Assistants, and offering proactive support when they are using the SoS. In addition, our approach considers robustness during SoS operation, trying to support it keeping coherent when constituent systems go down.

The work done until now allowed us to propose the MBA framework. We are currently developing a generic platform using this framework, as well as a method for building systems of systems. We started testing the approach on new problems,

in particular in the domain of health care (Wanderley et al. [6]).

ACKNOWLEDGMENT

Gregory Moro Puppi Wanderley would like to thank CNPq-Brazil (grant 233137/2014-9) for its support in this research.

REFERENCES

[1] L. v. Bertalanffy, *General system theory: Foundations, development, applications*. George Braziller, 1969.

[2] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of systems engineering: basic concepts, model-based techniques, and research directions," *ACM Computing Surveys (CSUR)*, vol. 48, no. 2, p. 18, 2015.

[3] G. M. P. Wanderley, M.-H. Abel, J.-P. Barthès, and E. C. Paraiso, "A core architecture for developing systems of systems," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 141–146.

[4] S. Hadian and K. Madani, "A system of systems approach to energy sustainability assessment: Are all renewables really green?" *Ecological Indicators*, vol. 52, pp. 194–206, 2015.

[5] S. Ford, U. Rauschecker, and N. Athanassopoulou, "System-of-system approaches and challenges for multi-site manufacturing," in *System of Systems Engineering (SoSE), 2012 7th International Conference on*. IEEE, 2012, pp. 1–6.

[6] G. M. P. Wanderley, É. Vandenbergh, M.-H. Abel, J.-P. A. Barthès, M. Hainselin, H. Mouras, A. Lenglet, M. Tir, and L. Heurley, "Consignela: A multidisciplinary patient-centered project to improve drug prescription comprehension and execution in elderly people and parkinsonian patients," *Telematics and Informatics*, 2017.

[7] L. Bai, "System of systems engineering and geographical simulation: Towards a smart tourism industry information system," in *Advanced Communication Technology (ICACT), 2013 15th International Conference on*. IEEE, 2013, pp. 1015–1018.

[8] P. Hershey, M.-C. Wang, and D. Toppin, "System of systems for autonomous mission decisions," in *System of Systems Engineering (SoSE), 2013 8th International Conference on*. IEEE, 2013, pp. 129–134.

[9] J. P. Olivier, S. Balestrini-Robinson, and S. Briceno, "Approach to capability-based system-of-systems framework in support of naval ship design," in *Systems Conference (SysCon), 2014 8th Annual IEEE*. IEEE, 2014, pp. 388–395.

[10] J. C. Kilian and T. M. Schuck, "Architecture and system-of-systems design for integrated missile defense," in *System of Systems Engineering Conference (SoSE), 2016 11th*. IEEE, 2016, pp. 1–6.

[11] T. Friedrich, C. Hilbes, and A. Nordt, "Systems of systems engineering for particle accelerator based research facilities: A case study on engineering machine protection," in *Systems Conference (SysCon), 2017 Annual IEEE International*. IEEE, 2017, pp. 1–8.

[12] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998. [Online]. Available: http://dx.doi.org/10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D

[13] R. L. Ackoff, "Towards a system of systems concepts," *Management science*, vol. 17, no. 11, pp. 661–671, 1971.

[14] J. P. Dauby and S. Upholzer, "Exploring behavioral dynamics in systems of systems," *Procedia Computer Science*, vol. 6, pp. 34–39, 2011.

[15] C. Stary and D. Wachholder, "System-of-systems supporta bigraph approach to interoperability and emergent behavior," *Data & Knowledge Engineering*, vol. 105, pp. 155–172, 2016.

[16] M. Jamshidi, *Systems of systems engineering: principles and applications*. CRC press, 2017.

[17] M. W. Maier, "Architecting principles for systems-of-systems," *INCOSE International Symposium*, vol. 6, no. 1, pp. 565–573, 1996. [Online]. Available: http://dx.doi.org/10.1002/j.2334-5837.1996.tb02054.x

[18] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, and T. M. Shortell, *Systems engineering handbook: A guide for system life cycle processes and activities*. John Wiley & Sons, 2015.

[19] B. P. Zeigler, S. Mittal, and X. Hu, "Towards a formal standard for interoperability in m&s/system of systems integration," in *GMU-AFCEA Symposium on Critical Issues in C4I*, 2008.

[20] A. M. Madni and M. Sievers, "System of systems integration: key considerations and challenges," *Systems Engineering*, vol. 17, no. 3, pp. 330–347, 2014.

[21] D. A. DeLaurentis, "A taxonomy-based perspective for systems of systems design methods," in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 86–91.

[22] S. Agarwal, L. E. Pape, C. H. Dagli, N. K. Ergin, D. Enke, A. Gosavi, R. Qin, D. Konur, R. Wang, and R. D. Gottapu, "Flexible and intelligent learning architectures for sos (fila-sos): Architectural evolution in systems-of-systems," *Procedia Computer Science*, vol. 44, pp. 76–85, 2015.

[23] M. R. Cutkosky, R. S. Engelmore, R. E. Fikes, M. R. Genesereth, T. R. Gruber, W. S. Mark, J. M. Tenenbaum, and J. C. Weber, "Pact: an experiment in integrating concurrent engineering systems," *Computer*, vol. 26, no. 1, pp. 28–37, Jan 1993.

[24] H. Park, J. Tenenbaum, and R. Dove, "Agile infrastructure for manufacturing systems: A vision for transforming the us manufacturing base," in *Proceedings of the Defense Manufacturing Conference*, 1993.

[25] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on computers*, no. 12, pp. 1104–1113, 1980.

[26] J.-P. A. Barthès, "Flexible communication based on linguistic and ontological cues," in *E-Technologies: Transformation in a Connected World - 5th International Conference, MCETECH 2011, Les Diablerets, Switzerland, January 23-26, 2011, Revised Selected Papers*, 2011, pp. 131–145. [Online]. Available: https://doi.org/10.1007/978-3-642-20862-1_9

[27] M. Fuckner, J.-P. A. Barthès, and E. E. Scalabrin, "Using personal assistant dialogs for automatic web service discovery and execution," in *WEBIST 2013 - Proceedings of the 9th International Conference on Web Information Systems and Technologies, Aachen, Germany, 8-10 May, 2013*, 2013, pp. 189–198.

[28] A. Jones, A. Kendira, C. Moulin, J.-P. A. Barthès, D. Lenne, and T. Gidel, "Vocal interaction in collocated cooperative design," in *Cognitive Informatics & Cognitive Computing (ICCI* CC), 2012 IEEE 11th International Conference on*. IEEE, 2012, pp. 246–252.

[29] M.-H. Abel, "Knowledge map-based web platform to facilitate organizational learning return of experiences," *Computers in Human Behavior*, vol. 51, pp. 960–966, 2015.

[30] J.-P. A. Barthès, "Omas a flexible multi-agent environment for cscwd," *Future Generation Computer Systems*, vol. 27, no. 1, pp. 78–87, 2011.

[31] J. Pérez, J. Díaz, J. Garbajosa, A. Yagüe, E. Gonzalez, and M. Lopez-Perea, "Towards a reference architecture for large-scale smart grids system of systems," in *Proceedings of the Third International Workshop on Software Engineering for Systems-of-Systems*. IEEE Press, 2015, pp. 5–11.

[32] P. Varga, F. Blomstedt, L. L. Ferreira, J. Eliasson, M. Johansson, J. Delsing, and I. M. de Soria, "Making system of systems interoperable–the core components of the arrowhead framework," *Journal of Network and Computer Applications*, vol. 81, pp. 85–95, 2017.

[33] R. K. Wong, C. H. Chi, Z. Yu, and Y. Zhao, "A system of systems service design for social media analytics," in *Services Computing (SCC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 789–796.

[34] F. Oquendo, "Formally describing the software architecture of systems-of-systems with sosadl," in *System of Systems Engineering Conference (SoSE), 2016 11th*. IEEE, 2016, pp. 1–6.

[35] C. Ingram, R. Payne, and J. Fitzgerald, "Architectural modelling patterns for systems of systems," in *INCOSE International Symposium*, vol. 25, no. 1. Wiley Online Library, 2015, pp. 1177–1192.

[36] B. Ge, K. W. Hipel, K. Yang, and Y. Chen, "A novel executable modeling approach for system-of-systems architecture," *IEEE Systems Journal*, vol. 8, no. 1, pp. 4–13, 2014.

[37] J. S. Dahmann and K. J. Baldwin, "Understanding the current state of us defense systems of systems and the implications for systems engineering," in *Systems Conference, 2008 2nd Annual IEEE*. IEEE, 2008, pp. 1–7.

[38] J. Dahmann and G. Roedler, "Moving towards standardization for system of systems engineering," in *System of Systems Engineering Conference (SoSE), 2016 11th*. IEEE, 2016, pp. 1–6.