Collaborative Networked Organizations as System of Systems: A Model-Based Engineering Approach

Mustapha Bilal, Nicolas Daclin, and Vincent Chapurlat

LGI2P, Laboratoire de Génie Informatique et d'Ingénierie de Production,
ENS Mines Alès, Parc Scientifique G. Besse, 30035 Nîmes Cedex 1, France
{Mustapha.Bilal,Nicolas.Daclin,Vincent.Chapurlat}@mines-ales.fr

Abstract. It is admitted that there is parallel between a System of Systems (SoS) and Collaborative Networked Organizations (CNOs). SoS Engineering (SoSE) carefully focuses on choosing, assembling and interfacing existing systems to build the so-called SoS. In this context, and as demonstrated by the literature and the System Engineering domain, interoperability takes on its full meaning and has to be fully considered as a decisive factor when organizations set up a CNO. This paper proposes to 1) model the SoS through a meta-model that includes concepts which 2) enable interoperability modeling and the analysis of its impact on the SoS' characteristics, stability, integrity, performance and behavior. The proposed analysis is based on a verification approach mixing simulation and formal proof techniques.

Keywords: Collaborative Networked Organization (CNO), System of Systems (SoS), System of Systems Engineering (SoSE), Interoperability, Modeling, Verification.

1 Introduction

Today's organizations tend to set up their Collaborative Networked Organization (CNO) by integrating more complex and capable entities. This collaboration helps to overcome the limitations of each organization to achieve a common mission that an organization alone cannot achieve [1]. In this sense, a CNO can be considered as a System of Systems (SoS) in terms of the ARCON reference modeling framework [2]. Hence, System of Systems Engineering (SoSE) started to be a critical need to conduct such complex and large systems design in the context of sociotechnical environments. SoSE can be distinguished from the classical System Engineering (SE) [3]. Beyond the classical SE, SoSE carefully focuses on choosing first the entities (i.e., existing, pre-defined entities) that will form the SoS, then assembling and interfacing them to allow their interaction. In this context, the interoperability of each entity takes on its full meaning and has to be considered prior the assembling. Assembling the entities establishes some kind of interactions between them. This imposes to define and validate interfaces of various types (e.g. physical, procedural, etc.) to facilitate and permit these interactions. In other words, defining the interfaces allows entities to improve their capacities to work together, i.e. to enhance their interoperability,

L.M. Camarinha-Matos and H. Afsarmanesh (Eds.): PRO-VE 2014, IFIP AICT 434, pp. 227–234, 2014.
© IFIP International Federation for Information Processing 2014

without huge efforts or reverse effects. The requested interactions between socio-technical entities are not been yet addressed by the literature. It is a new challenging area and there is a trend toward discovering it [4]. Therefore, it is required to enable to manage various kinds of interactions between the entities.

This paper focuses first on the similarity between SoS and CNO. Afterwards, it analyzes the relationship between the interoperability and other SoS' characteristics. It contributes then to the field of the SoSE by introducing a means to better understand the interoperability's impact on the SoS' analysis perspectives defined here as stability, integrity and performance as proposed in [5]. The first objective is to develop a Domain Specific Modeling Language (DSML) that allows having a working environment to model the SoS and analyze the SoS model in a coherent way. This DSML permits to consider the SoS dynamicity and to describe SoS' global architecture considering interoperability and analysis perspectives dimensions. The second objective is to assist managers, engineers and designers, involved in SoSE process in achieving verification tasks and particularly to check the relationships between interoperability and the other SOS' characteristics described below. This is based on a verification approach applied on the SoS model and on complementary formalisms (formal proof of properties and simulation techniques).

2 SoSE Principles

SoS vs. CNO - Giving a definition for the SoS and CNO is required to draw a parallel between these two concepts. The literature provides a large number of definitions given to the SoS. However, this paper will not delve into the current debate of choosing an appropriate definition. Thus, and in attempt to come to terms with these definitions, it is largely agreed that a SoS is seen as a group of, in most cases, existing entities assembled together to interact, during a timeframe, to produce some kind of capabilities, products or services and to **achieve a global mission** that a system alone cannot fulfill [6]. Moreover, there are seven crucial characteristics for the SoS [7][8], such as: **Operational Independence**, **Managerial Independence**, **Evolutionary Development**, **Emergent Behavior**, **Geographic distribution**, **Connectivity** and **Diversity**. SoS' size, its complexity and its characteristics, induce an additional effort over the SE [3] to respond to what it is expected from the designed SoS.

On the other hand, a collaborative network is "*a network consisting of a variety of entities (e.g. organizations and people) that are largely* ***autonomous, geographically distributed****, and* ***heterogeneous*** *in terms of their operating environment, culture, social capital and goals, but that* ***collaborate*** *to better* ***achieve common or compatible goals****, thus jointly generating value, and whose* ***interactions*** *are supported by computer network*" [1]. Furthermore, most CNs require an organization over the activities of their entities, fixing roles for the participants, and some governance rules. Therefore, these CNs can be considered as manifestations of CNOs. The hereinbefore definitions highlight the first similarities between some SoS and CNOs fundamental characteristics. Furthermore, both SoS and CNO life cycles draw the second line of similarities. At macroscopic level, a CNO life cycle comprises at least four phases known as Creation, Operation, Evolution, Dissolution, or eventually Metamorphosis [1]. During the Creation phase (corresponding to the Assembling

phase of the SoS), the entities that will form the CNO are selected depending on their capacity to participate in performing the global mission and their abilities to interoperate. In the Operation phase (Connectivity phase of the SoS), entities are supposed to be able to exchange, cooperate and interact together. In the Evolution and Metamorphosis phases (Evolution phase of the SoS), internal and external changes occur, e.g. adding, removing, updating entities, or even when major changes in objectives or principles take place. In Dissolution phase, CNO ceases to exist and each entity returns to its own mission.

Interoperability vs. CNO Analysis Perspectives - Two entities are interoperable if they are capable of establishing a link that allows them to perform a mission and to destroy it once the mission is accomplished. We argue that there is a strong linkage between interoperability and the CNO's basic characteristics. This link concerns the CNO's life cycle and all CNO's entities for the following reasons. First, each CNO's entity can evolve separately from the other entities, therefore CNO's entities have to be capable of building links with each other among their interfaces and destroying them dynamically [8]. Furthermore, it becomes possible to easily remove, modify or add an entity. This **dynamic evolution** is consistent with the loosely coupling hypothesis offered by interoperability. Second, entities are of **heterogeneous** nature (notion of diversity) but have effectively to be able to exchange with each other without enormous effects. This is consistent with the 'without interfacing effort" (plug and play) offered by interoperability. Last, entities must stay **independent** in an operational sense (they continue to provide flows and services corresponding to their own mission) and managerial sense (they remain able to take decisions). Once more, this is consistent with the hypothesis of **autonomy** offered by interoperability [9].

A lack of interoperability of one or several CNO's entities can impact the CNO's analysis perspectives:

- **Stability** is the quality that reflects the CNO ability to maintain its viability and to adapt to any change in its environment [5] (e.g. adding a new entity) by returning to a previous known and controlled functioning mode. A CNO shows both homeostasis and adaptive behavior, as it can constructively respond to disturbances or novel environmental conditions. Therefore, improving the interoperability helps to ensure and to increase the CNO stability all over its lifetime even if none of the existing architectural styles of self-adaptation for SoS guarantee the SoS' stability [10].
- **Integrity** is the quality that reflects the CNO ability to maintain its viability and to adapt to any change in its environment when facing a local modification e.g. modifying or removing one of the existing entities. If an entity inside a CNO is interoperable then it is able to pursue the collaboration without huge impact on the global behavior. In the same way, if this entity has to be replaced, then the requested interoperability level of the new one guarantees a well-functioning of the whole CNO.
- **Performance** is the quality that reflects the CNO ability to reach its performance's objectives. The goal here is not to guarantee a maximum level of performance but to become able to meet more rapidly a sufficient level of performance. For this, the 'plug and play' vision, permitted by an interoperable entity, seems more interesting because it allows reducing time, costs and efforts requested in case any changes in the CNO's entities take place (adding, leaving, modifying).

SoSE Needs and Work's Contributions - Considering the similarities between CNO and SoS, it seems hazardous to build a CNO without engineering approach such as proposed for SoS and named SoSE. The subject of SoSE versus SE is debated in the literature. Some authors agreed that SE principles, processes and standards are enough to perform SoSE activities [11] and no additional processes are needed. However, SoS characteristics, assembling, interfacing and interactions between its entities, induce an additional effort over the SE [3]. SoSE is classically considered as a model-based approach. A model helps to address new requirements, presents a better understanding of the SoS' entities and their relationships, it helps to understand the SoS functionality and monitors and assesses changes all over SoS evolution. Therefore, as in any other scientific or engineering discipline, the first need is to propose relevant modeling languages to better understand the SoS area and to supervise and manage the operations of the SoS during its life cycle. Furthermore, a model should help to predict and simulate behaviors and offer a means for better decision-making. The existing modeling methods do no cover all the needs of CNOs and there is no single formal modeling approach to model all CNOs problems [12]. Therefore, this research addresses the first SoS need by proposing a 1) meta-model that groups all concepts and relations required to model various kinds of SoS (CNO). Moreover, decision-making in all SoS' lifecycle should be based on well verified models. Therefore, 2) verifying the SoS model is the second SoS' need.

3 SoS Modeling Languages

The modeling phase contains, on the one hand, a tailoring of existing and traditional SE modeling languages focusing on requirements, functional and physical modeling. On the other hand, this phase requires having a means and languages for describing behavioral aspects and particularly to describe interactions between the SoS' entities (see Fig. 1). Available DSMLs do not allow to model seamlessly and homogeneously the four views of the SoS. The here developed DSML offers a unique and homogenous environment that allows at the same time, (1) to model the SoS, and (2), to analyze the impact of the interoperability on the SoS analysis perspectives.

Requirements Modeling Language (ML): Requirement engineering activities for a SoSE are greatly expanded. To understand the SoS requirements, they should be modeled at different levels and from three viewpoints. The first viewpoint is the users' perspectives (stakeholders' requirements). The second viewpoint covers the entities' requirements. Finally, and at the highest level, the third viewpoint which is the SoS requirements / characteristics that have to be respected and maintained all over its life cycle. Moreover, during SoSE and further to the traditional "*-ilities*", new ones, such as adaptability, **interoperability**, flexibility etc. are imposed.

Physical ML: In SoS, it is important to understand the set of entities which enable the SoS capability and to understand how these entities are interfacing together to interact and contribute to the SoS objectives. Interfacing the entities appears in the creating phase of the CNO life cycle. One of the famous problems in CNO is the physical integration of multiple entities due to the diversity of interfaces [13]. Assembling the entities establishes some kind of interactions which make them able to work together. These interactions impose to have interfaces of various type:

technical (respecting general standard of physical interconnection of technical systems, software etc.), physical (hardware), informational (model, knowledge, information and data exchange protocols), organizational (communication rules, separation process public/private, protocols and rules of: organization, control, taking responsibility, delegation etc.) or HMI (human machine interface). These interfaces are necessary to ensure entities' interoperability. The challenge raised here is to design the interfaces which will improve the interoperability by managing the interactions without affecting the entities. This phase will facilitate the verification phase where a part of it has to be done on the interfaces between the SoS' entities.

Functional ML: CNO function is the collection of coherent processes, which perform in a continuous way within the CNO, to support the CNO's mission. SoS functional architecture is the premier key for the SoS' success. Its architecture should resemble the entities and SoS functions. CNO life cycle helps to draw some functions to consider in the functional architecture. The minimal life cycle of CNO comprises four phases and, during these phases, new functionalities and requirements are elicited. These functionalities are ensured by a set of core elements of the here-proposed SoS model. The functional concepts of the "Creation" phase cycle of the meta-model do not fully deal with the partner's selection. Multiple researches contributed to the partner's selection during the assembling phase of the SoS [14]. However, this SoS model facilitates this selection by analyzing their aptitude and capacity required to provide the services/products/resources which allow them to participate in the SoS's global mission. Moreover, this model offers a means to select the partners' with compatible interfaces adapted to allow interoperability.

Behavioral ML: The interactions between entities are important to fulfill the SoS' mission, or also to prevent and avoid any identified or even identifiable disturbing events or risky situation. However, these interactions can be at the origin of various emergent behaviors and properties that remain not predictable. Therefore, it is essential to model them to improve the local interoperability between entities and to allow controllability of their impact on the SoS. The interactions between the entities of the CNO appear mainly in the operational phase of the CNO life cycle. This phase includes some functionalities such as: Basic information exchange interactions, Events/exception handling, Advanced cooperation, Material/services related aspects and collaborative environments [15]. The partners/entities must ensure that the interactions between them are effective and, eventually, that their autonomy (operational and governance autonomy – managerial and operational independency) is preserved. An *interaction* is an emission of a source entity - *SoSEntity* (materialized by a *Flow* or *Field*), that impacts directly or indirectly the state of one or more destination entity - *SoSEntity* leaving a concurrent/non-concurrent *Effect* on each of these destinations. An *Effect* is the embodiment of an *Interaction*. It is the result manifested by the impact on the structure, mission and/or behavior of a *SoSEntity* in terms of analysis perspectives (Stability, Integrity and Performance). An *Interaction* is caused by a reaction, a feedback, *Event* or a chain of events. An *Event* announces a *Risk* which is produced only and only if there is a *Danger* and *Vulnerability*. A *Danger* is caused by an *Event* or a combination of events. Moreover, a *Risk* can have serious impacts on the mission of the SoS (functional aspect - performance) and/or its structure (organic aspect- *SoSEntity* - costs) and consequently on the system usage.

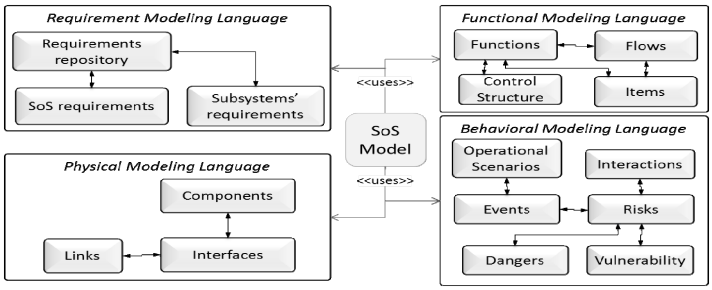

Fig. 1. Main concepts and relations modeling elements for building SoS model

4 Verifying the SoS Model

Verifying the SoS model, whatever may be its size and complexity, is not being yet fully discovered by the research. The goal here is to check that the SoS' requirements have been taken into account when building the SoS model. Some of these requirements focus on the interoperability. The proposed verification technique intends to mix execution and formal proof techniques. The execution allows simulating the SoS' model behavior and formal proof techniques allow proving properties. Indeed, requirements are being formalized under the form of provable properties. As described in Fig. 2, the execution consists first on launching one modeled operational scenario which describes how a SoS may evolve in a particular context. A set of *D*isturbances *D* is defined according to a repository that includes types of disturbances such as adding new entities, retrieving entities, modifying entities, etc. For a chosen scenario, a set of interoperability requirements to analyze is fixed (phase 1) *e.g.* an acknowledgement is requested after an exchange of flows between entities [16]. The simulation is then launched. With each instant *T* a disturbance D occurs (phase 2), the simulation is stopped and the SoS model state (*S*) is frozen (phase 3). The selected properties are then formally proven starting from the current state *S* (phase 4) and ending with the set of future reachable SoS model states (*S'*). S' are obtained by using an execution path finding algorithm starting from the initial S. Moreover, S' might be fully independent of the initial operational scenario. The formal proof is done by using a model checking technique [17]. The goal is to detect changes in terms of requirements that are not checked at a specific moment. Furthermore, these changes can be detected for all future states that become reachable due to the behavioral interactions between the SoS' entities. Any modification in the interoperability requirements induces a variation on the SoS' analysis perspectives. As presented in Section 2, if the interoperability is improved, the SoS integrity and stability are improved due to the notion of "plug-and-play" of the interoperability. However, the SoS performance might increase or decrease. For example, if the frequency of the information exchange rate between the SoS' entities increases, theoretically this induces an improvement in the SoS performance, however, if any entity cannot/not ready to absorb the increased frequency, this will induce a decrease in the performance. In other words, if an enterprise A is sending products to another enterprise B in the same CNO at a frequency F, if F increases, the SoS performance

increases since the production/sending frequency is now higher, however, if B does not have the enough capacity to receive/stock the products at this frequency, this induces some kind of system overload and the SoS performance will then decrease. Therefore, this kind of variation has to be interpreted on the model by an expert after fixing him a set of rules which allows him to make the correct interpretation. The simulation will resume until every possible disturbance is injected (phase 5).

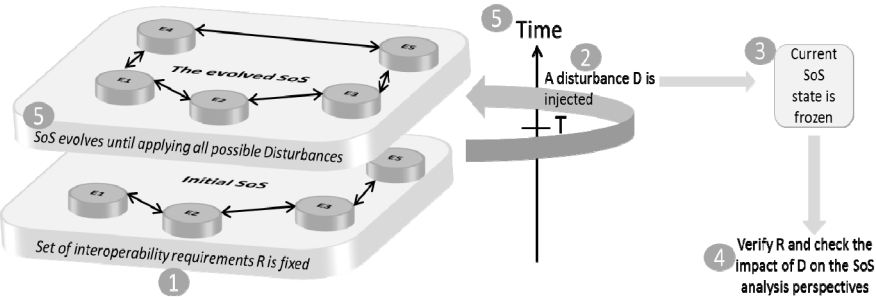

Fig. 2. Verification methodology mixing simulation and formal proof techniques

Aspect Waving Model approach based on a meta-programming environment e.g. Kermeta [18] is used for simulation. The advantage of using the Aspect Waving Model approach is to avoid the data loss and time consuming resulted from transforming the modeling language into other language (e.g. Multi-Agent language). It is possible, by using this approach, to recover the developed model without any transformation, to add properties on the fly (attributes and methods) and to describe the behavior of each method (by giving it an operational semantic). However, the use of the Model Checking requires having a deterministic SoS behavioral model. Therefore, a step-by-step simulation has been used here where it is possible to have a deterministic SoS behavioral model at a definite instant of the simulation.

5 Conclusion and Perspectives

CNO and SoS share the same crucial characteristics. Therefore, SoSE is used to guide the CNO. Interactions between subsystems are required to achieve SoS' global mission. However, these interactions impose to have interoperable systems. This paper has shown how interoperability shares common characteristics with the SoS and how it impacts the SoS' analysis perspectives. A SoS model has been presented. It describes the SoS behavior and the interaction between its entities. A verification approach based on formal proves and simulation has been proposed. It helps to understand the interoperability's impact on the SoS' analysis perspectives.

We aim to propose an interoperability requirements repository and to define an operational semantic for the proposed DSML to make a model provable and interpretable i.e. to permit the simulation of the SoS behavior taking into account its environment. We are willing to verify the consistency of the developed meta-model, refine it with the new interoperability requirements, verify that all the interaction have been considered and finally check that the SoS' analysis perspectives remain relevant.

References

1. Camarinha-Matos, L.M., Afsarmanesh, H., Galeano, N., Molina, A.: Collaborative networked organizations – Concepts and practice in manufacturing enterprises. Comput. Ind. Eng. 57, 46–60 (2009)
2. Camarinha-Matos, L.M., Afsarmanesh, H.: Collaborative Networks: Reference Modeling. Springer (2008)
3. Blanchard, B.S., Fabrycky, W.J.: Systems Engineering and Analysis (2011)
4. Camarinha-Matos, L.M.: Collaborative networks: A mechanism for enterprise agility and resilience. In: Enterp. Interoperability VI, pp. 1–8 (2014)
5. Bilal, M., Daclin, N., Chapurlat, V.: System of Systems design verification: problematic, trends and opportunities. In: Enterprise Interoperability VI, pp. 405–415. Springer International Publishing (2014)
6. DeLaurentis, D., Callaway, R.K.: "Cab:" A System-of-Systems Perspective for Public Policy Decisions. Rev. Policy Res. 21, 829–837 (2004)
7. Maier, M.W.: Architecting principles for systems-of-systems. Syst. Eng. 1, 267–284 (1998)
8. Stevens Intitute of Technology, Castle Point on Hudson, Hoboken, N. 07030: Report on System of Systems Engineering (2006)
9. Chen, D., Daclin, N.: Framework for enterprise interoperability 1. In: Proc. IFAC Work. EI2N, pp. 77–88 (2006)
10. Weyns, D., Andersson, J.: On the challenges of self-adaptation in systems of systems. In: Proc. First Int. Work. Softw. Eng. Syst. - SESoS 2013, pp. 47–51 (2013)
11. Clark, J.O.: System of Systems Engineering and Family of Systems Engineering from a standards, V-Model, and Dual-V Model perspective. In: 3rd Annu. IEEE Syst. Conf. (2009)
12. Camarinha-Matos, L.M., Afsarmanesh, H.: A comprehensive modeling framework for collaborative networked organizations. J. Intell. Manuf. 18, 529–542 (2007)
13. Reithofer, W., Naeger, G.: Bottom-up planning approaches in enterprise modeling-the need and the state of the art. Comput. Ind. 33, 223–235 (1997)
14. Mikhailov, L.: Fuzzy analytical approach to partnership selection in formation of virtual enterprises. Omega 30, 393–401 (2002)
15. Camarinha-Matos, L.M., Afsarmanesh, H.: Virtual enterprise modeling and support infrastructures: Applying multi-agent system approaches. In: Luck, M., Mařík, V., Štěpánková, O., Trappl, R. (eds.) ACAI 2001. LNCS (LNAI), vol. 2086, pp. 335–364. Springer, Heidelberg (2001)
16. Mallek, S., Daclin, N., Chapurlat, V.: The application of interoperability requirement specification and verification to collaborative processes in industry. Comput. Ind. 63, 643–658 (2012)
17. Bérard, B., et al.: Systems and Software verification: model checking techniques and tools (2010)
18. Muller, P.-A., Fleurey, F., Jézéquel, J.-M.: Weaving executability into object-oriented meta-languages. In: Briand, L.C., Williams, C. (eds.) MoDELS 2005. LNCS, vol. 3713, pp. 264–278. Springer, Heidelberg (2005)