UNIVERSITE DE BLIDA1

Faculté des Sciences

Département d'Informatique

THESE DE DOCTORAT

MISSION ORIENTED PROCESS FOR SYSTEMS OF SYSTEMS ENGINEERING

Par

Imane CHERFA

Composition du jury:

H. Abed	Professeur	U. de Blida1	Présidente
N. Boustia	Professeur	U. de Blida1	Examinatrice
Z. Alimazighi	Professeur	U.S.T.H.B., Alger	Examinatrice
N. Chikhi	Maître de conférences	U. de Blida1	Examinateur
S. Sadou	Professeur	U. de Bretagne Sud, France	Dir. de thèse
D. Bennouar	Professeur	U. de Bouira	Dir. de thèse
N. Belloir	Maître de conférences	U. de Bretagne Sud, France	Encadrant

Blida, 30/05/2022

Abstract

Systems of Systems (SoSs) encompass a group of distributed and independent systems, which work together to achieve a common goal. Their design and development is becoming increasingly important in wide variety of application domains, such as trade, transportation, health-care and military. Interesting approaches were proposed in Model-Based System Engineering (MBSE) to tackle with the design, development and complexity management of SoSs. We cite as example the systems engineering (SE) model-based methodologies and the architecting frameworks. However, several aspects of SoSs as the operational independence of constituent systems (CSs) and the uncertainty of the SoSs environment may create additional challenges. This thesis addresses the research gap by proposing a Mission Oriented Process for System of Systems Engineering (MOP-SoSE), based on the System Modeling Language (SysML), that comprises: (1) a mission conceptual model that includes the concepts characterizing SoS mission and their relationships, (2) an utilization process to provide guidance on the use of our approach, and (3) an multi-agents simulation (MAS) steps to assess gaps in mission performance and to improve the architecture. To illustrate our approach we conducted a case study on crowd management SoS.

Keywords: Systems of Systems (SoSs), mission-oriented process (MOP), system modeling language (SysML), mission, model-based system engineering (MBSE).

Acknowledgements

I am deeply grateful to those who have supported and encouraged me on my path to completing this dissertation and would like to acknowledge their contributions here.

First of all, I would like to sincerely thank my supervisor, Professor Salah Sadou, that gave me the opportunity to work with him and to discover a nice research domain. Thanks to his help, I could start my path as a Ph.D. student and develop my work. He has encouraged me to work even harder and take on new challenges with his strong enthusiasm and passion for scientific research. Salah, thank you for your guidance, support, and comprehension.

I would like also to address my thanks to my supervisor, Professor Djamal Bennouar, for his belief in my abilities since I was a student. Thank you Professor Bennouar for your invaluable contribution to my academic and personal development, by being my mentor from graduating, Magistère, and now a Ph.D. I will always be grateful for the confidence you have shown in me throughout my journey.

I'm deeply indebted to my advisor, Doctor Nicolas Belloir. If I had the courage to continue and finish this thesis, it's because I had Nicolas by my side. I will never thank him for the guidance, help, patience, and motivation he gave me during the whole process, especially in difficult times. Nicolas has a nice balance of different traits: critical but supportive, candid but human, ambitious but understanding, strict but fair, respectful but caring and attentive. Nicolas, thanks for giving me the opportunity to work with you, and for your constant support, everything I could say would not be enough to thank you!!! I feel honored for your interest in my work.

I would like to extend my sincere gratitude to Régis Fleurquin, for spending time to review my work and to contribute to its improvement. His valuable comments, suggestions, and constructive and helpful remarks helped me significantly to improve my thesis.

My deepest appreciation goes to all the jury members, Professor Hafida Abed, Professor Narimène Boustia, Professor Zaya Alimazighi, and Doctor Nacim Chikhi, who agreed to be part of the committee. I am very pleased to have presented my Ph.D. work in front of such high qualified jury.

I gratefully thank my friends for their encouragement and support. Particularly, thanks to Imane Chikhi whose encouragement had a significant influence on this thesis. Rymel Benabidallah and Raounak Benabidallah, I cherish the memories of the times we spent together, I really enjoyed your company.

I want to thank my colleagues for their support and help, especially Nachida Rezoug, Imane Chikhi, Yasmine Mancer, Ratiba Zahra, Djawida Boumahdi, Messaouda Farah et Meriem Arkam. Their kind words, gestures, and unwavering support have kept me motivated. Many thanks to all my colleagues in the Computer Science department of the University of Blida1, I am honored to have worked alongside such kind, talented, and inspiring colleagues.

I would like to express my appreciation to all the outstanding staff members at IRISA Laboratory (UBS Vannes), who have always gone above and beyond to ensure that my internships were successful, especially Sylviane Boisadan.

I also think of the teachers I had, from primary school to university. Among them, I met many passionate and exciting people who encouraged me and taught me the desire to learn.

Last but not least, the last words are for my family members. I wish to dedicate this thesis to my family which has always been there for me with truthful support. I want to thank my parents who have always been my example, for their unconditional love. Thank you for sacrificing your life for us, for giving me the opportunity to do my studies in very good conditions, and for supporting, and encouraging me during all these years!! A big thank you to my unique sister Sihem for her love and her constant benevolence, for her husband for being like a brother to me, and for her adorable children who are like mine. I also want to thank my brothers Zakaria and Elyes who daily bring color to my life with their presence and affection. Thank you Elyes for receiving me at your home several times during my internships in France, and Raounak for making these internships fun and enjoyable. I would like to deeply thank my beloved aunts Ibtissem, Hayet, Karima et Wahiba who have always cherished me, and I would like to dedicate this thesis to my dear aunt Amal.

I want to thank my husband's family especially Naziha and Rafik for their support and all the members of my big family.

Finally, big thanks to my husband Amine, who sacrificed so much to help me finish my thesis, I am forever grateful. Thank you for leaving your job for a year to accompany me to France, for helping me manage my emotions, and for being an amazing technician who provided me with exceptional hardware and software troubleshooting. The hardest thanks to write are those of my children: Lyne, Djawad and Wassim, because the long hours of work meant that I had to sacrifice time with them. Thank you for being my pillars of strength, and my source of inspiration. I love you more than words can express.

Contents

List of Figures

List of Tables

INTRODUCTION

Context

Systems of Systems (SoSs) encompass a group of independent and distributed systems which, through synergism between them, work together towards a common mission (35, 36). The SoSs literature addresses a wide variety of application domains such as trade, transportation, healthcare and military (4). Several researchers listed various characteristics, to distinguish systems of systems from traditional systems (37, 38, 39). According to several authors (11, 40), the independence of the constituent systems (CSs) is the main feature of SoS (System of Systems). Each constituent system (CS) assumes its own goals and operates independently (41). The need to maintain autonomy while simultaneously operating within the SoS context considerably increases the complexity of an SoS and is at the heart of System Engineering (SE) (42).

To address the emerging generation of complex systems which includes SoSs, practitioners admitted the need to evolve the practice of SE. System of systems engineering (SoSE) represents the necessary extension and evolution of the traditional SE discipline, that allows to engineer SoSs and to manage complexity and change. Unlike systems, SoS could not be described in terms of hierarchies, but may be understood as an environment along with the constituent systems operate and interact within it (39).

To guide the selection of SoSE principles, researchers tried to devise SoSs into different types, based on their level of centralization into four categories: virtual, collaborative, acknowledged and directed (6, 10, 40, 43). These categories offer a framework for understanding them, to cope with the challenges arising from this class of systems. Furthermore, they influence how system engineering (SE) can be applied to SoSs in several aspects like management and oversight, operational focus, engineering and design considerations (10, 33). SoSE is primarily concerned with acknowledged

SoS, where the SoS is under the responsibility of an organization, a SoSE team that sustains the engineering of the SoS. Nevertheless, the Constituent Systems (CSs) of an SoS preserve their independent development and goals. So, the SoSE team does not have complete authority over them and changes in the CSs rely on collaboration between the CS team and the SoS team (6, 10).

Problem statement

In the literature, various interesting approaches were proposed that tackle with the design and development of SoSs. Model-Based System Engineering (MBSE) represents a promising path for the development and analysis of SoSs (36). Modeling allows SoS engineers to control the overall complexity of SoS, to reveal and document its structure and behavior, and to communicate these to stakeholders (44). A large number of model-based methodologies are currently used by the SE community, such as the Object-Oriented Systems Engineering Method (OOSEM) (14), Harmony (15, 45) and MagicGrid (16), which are dedicated to analyze, developing and documenting complex systems. But when using such methodologies, SE engineers should be able to trace system boundaries and define requirements clearly. Furthermore, they should master the development environment to assure that the technical trade studies are the basis of the allocations of requirements to components. In the SoS environment, SE engineers must take into account considerations beyond the use of existing systems as CSs. They must allocate the realization details and functionalities, which may not be optimal from the point of view of the SoS. Furthermore, CSs must retain their independence. For these reasons, SE is highly challenging in the SoS context.

On the other hand, other research has focused on expressing the SoS within architecture frameworks (AF) to manage design complexity. This is done by the use of several layers named architecture views to represent the system from different viewpoints (for example operational, functional, or system). The United States Department of Defense Architecture Framework (DoDAF) (46), the British Ministry of Defense Architecture Framework (MODAF) (47) and the North Atlantic Treaty Organization (NATO) [1] Architecture Framework (NAF) (48) are the most often used architecture

[1]An intergovernmental military alliance between several countries to support collective security. Web site: https://www.nato.int Accessed 15/11/2021

frameworks for modeling SoSs. However, these frameworks do not provide step-by-step methodological guidance to be followed when using them to analyze or design SoSs. Moreover, they do not offer a precise connection between the different views of the SoS.

Several solutions intended to provide and evaluate model-based methods and tools for development and analysis of SoSs have been proposed in the context of research projects. Comprehensive Modeling for Advanced System of Systems (COMPASS) (49), Designing for Adaptability and EvolutioN in System of systems Engineering (DANSE) (9) and Architecture for Multi-criticality Agile Dependable Evolutionary Open System of Systems (AMADEOS) (20) are examples of them. Each solution brings interesting aspects for SoSE. The main advantage of the COMPASS approach is that it provides a well-defined denotational semantic of the architecture diagrams, which allow to support a variety of analysis techniques. DANSE proposed methodological guidance and a reduction in the AF according to the target objectives while AMADEOS offers an integrated support to all the architecture viewpoints. It offers means to link the high-level perspective to activities and artifacts involved in SoS design phases. However, more recent research argues that in SoSE domain, the design and the end-to-end process and management are required to be balanced (22, 50). The end-to-end process is a key element to assist SoS engineers to determine the systems that must be involved and the functions they must perform. It is important to consider the end-to-end-process to bridge the dissociation between the SoS objectives and the individual functionalities undertaken by the CSs.

The state of the art of SoSs modeling reveals that SoSE still lack of life-cycles, processes, engineering activities, and tools to analyze, design, architect, simulate and evolve SoSs.

Challenges

From the aforementioned problems, we identify five challenges for developing SoSs:

- Evolving life-cycle: evolutionary development characterizes SoS. Thus, every process dedicated to an SoS should consider the evolutionary aspect of SoS.

- Constituent systems independence: the engineering of constituent systems is done independently with the engineering of the SoS. Every process must differentiate between the two levels of engineering.

- Stakeholders involvement: every process must distinguish the two levels of stakeholders (constituent system and SoS level). Each of these stakeholder groups has its own goals and organizational circumstances, which affect their expectations for the SoS. The constraints and development plans for constituent systems may be unknown to the SoS's stakeholders.

- Uncertainty of the SoSs environment: SoSs environment is uncertain because the constituent systems available at run-time are not really known in the early stages of SoS development. The SoS manager does not have complete control over all of the constituent systems that affect the SoS capabilities.

- Function allocation to CSs: allocation of functions or activities to CSs depends closely on the end-to-end-process. Thereby, it is important to consider it to bridge the dissociation between the SoS objectives and the individual functionalities undertaken by the CSs to support the objectives.

Contributions

We argue that the end-to-end process is embodied in the concept of mission. In fact, Mission Engineering (ME) is the area that intends to link the engineering activities that are conducted to achieve a mission, with the mission itself (51). Recently, the ISO/IEC/IEEE 21839 (52) has stressed the importance of considering the mission, and its context, in the development life-cycle of systems. ME goal is to address the end-to-end mission as System of Interest (SOI). A mission has a goal, which is achieved through a sequence of operational activities. ME determines those operational activities and allocates them to operational nodes for execution. Therefore, we propose in this thesis to put the mission in the heart of the SoSE analysis and the architecture process. In our case, the operational activities are allocated to different CSs of the SoS.

Thus, we propose in our thesis a process to build SoSs, which combine and take advantage from ME and MBSE, called MOP-SoSE (Mission Oriented Process for System of Systems Engineering). The key ideas on which the process relies are as follows: (i) The process is applicable to acknowledged SoS, in which the organization manages the SoS, and supports the SoSE. Independent organizations and SE teams are responsible for the constituent systems. (ii) An SoS is considered as an environment within which CSs operate, to accomplish a given mission. (iii) Mission context determines mission thread, and then mission context help to determine the functionalities needed, and then the CSs to be involved. (iv) The end-to-end mission is considered in our process as SOI, from which architecture is generated as automatically as possible, to avoid information loss between the application domain expert and the system architect. (v) The concrete architecture is elaborated from the abstract one. The latter serves as an invariant that guides the choices of concrete entities.

To our knowledge there is no existing approaches that support all the challenges raised by the problem pointed above. To tackle these challenges, the present thesis provides the following contributions:

1. **The proposition of mission conceptual model that includes the concepts characterizing SoS mission and their relationships**, based on the literature review on SoSE and ME. The conceptual model served as the basis for our process and helped to draw a link between the mission model constructs and architectural constructs.

2. **Development process based on mission modeling to build SoSs architecture:** this development process includes the engineering activities that allows SoS engineers to refine the mission into architecture. The role of participants in such a process is described clearly.

3. **The proposition of simulation steps to evaluate and improve the SoS effectiveness:** the multi-agents simulation is based on the architectural models. It is performed in order to collect the different effectiveness measures to evaluate the architecture, detect deficiencies and implement updates.

Structure of the document

This document is structured in six chapters: The first three ones constitute the state of the art which provides a foundation to understand the contribution.The next two chapters detail the contributions of this thesis, while the last chapter gives a conclusion to this work.

Here is a short description of each chapter:

Chapter 1 (system of systems engineering background) starts with an overview of SE basic concepts. It presents then the foundation, concepts and characteristics of SoSs. It compares SE and SoSE domains and finally highlights SoSE key challenges.

Chapter 2 (system of systems modeling) discusses the state of the art in SoSE and modeling. It lists the SoSE life-cycle models, and SoSE architecting artifacts. It gives then a detailed presentation of the SoSE processes, engineering activities and AF which are used in SoSE. The remainder of the chapter illustrates the research gaps in the state of the art of SoSE.

Chapter 3 (mission engineering) presents an overview of the ME domain and provides the definition of involved concepts and roles. Furthermore, different ME approaches are listed. The chapter ends with a discussion that determines the limitations of existing approaches in order to position the contributions of the thesis.

Chapter 4 (mission-oriented process for system of systems engineering) presents a development process to build SoSs. It is based on the combination of ME concepts and MBSE fundamentals. The process's life-cycle, roles, concepts, and engineering activities are detailed in this chapter. The resulting models are expressed using the System Modeling Language (SysML) and Emergency Response System of systems (ERSoS) is used as an illustrative example to illustrate the different steps.

Chapter 5 (case study modeling and simulation) shows how our process can be applied using crowd management SoSs case study. Secondly, we present the simulation step as well as its real implementation.

Conclusion and perspectives highlights the contributions of the thesis and gives some perspectives and future work.

Chapter 1

SYSTEMS OF SYSTEMS ENGINEERING BACKGROUND

System Engineering (SE) has been around for hundreds of years, but it was only in the 1950s that it became a recognized field (53). The expanding space, missile, and nuclear weapon races at the period needed a higher quality of system development, integration, and testing than previously. SE procedures made their way into the world of industry not long after. As systems became more complex and network-centric, the paradigm of a System of Systems (SoS) emerged, including that complex set of distributed, independent, and interacting systems (37). The need to manage this class of systems leads System of Systems Engineering (SoSE) to be born. Thus, this chapter offers an introduction of the discipline of SoSE and highlights the main differences between SE and SoSE. It begins with a few primary definitions related to the system concept, a short survey of the SoSE discipline's history, list definitions, characteristics, and challenges in the SoSE domain as well as the different topologies of SoS. The chapter ends by addressing the main differences between SE and SoSE based on different perspectives.

1.1 Definitions and basic concepts of a system

In this section, we give the definition of the fundamental concepts of monolithic systems and thus of the SE domain. We discuss the apparition of SoSE domain to deal with system complexity.

1.1.1 System concepts

The ISO/IEC/IEEE 15288 (54) and the systems engineering handbook (1) consider that systems in the real world are "man-made, created and utilized to provide products or services in defined environments for the benefit of users and other stakeholders". Thereby, the system concept definition should distinguish between systems in the real world and system "mental representations". From this view of a system, the INCOSE and ISO/IEC/IEEE define the system concept as follows:

"... an integrated set of elements, subsystems, or assemblies that accomplish a defined objective. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, services, and other support elements" (1).

"... combination of interacting elements organized to achieve one or more stated purposes" (54).

The systems engineering handbook highlights that these definitions share one main idea, that a system is a "purposeful whole that consists of interacting parts", as shown in Figure 1.1. We are going to rely on this idea throughout this manuscript. In what follows, we present definitions of key concepts closely related to the concept of system.

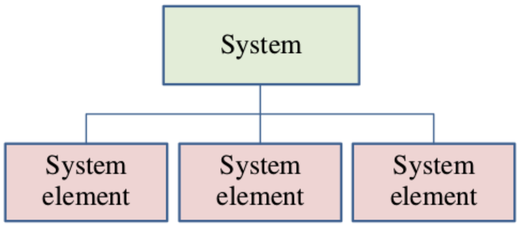


Figure 1.1: System composed of interacting elements (1)

The system **context or operating environment** is the external view of a system that includes set of elements which do not belong directly to the system but interfere with it. Users (or operators) of the system are examples of system context (1).

The system **boundary** concept takes place to differentiate a system's internal view from its external view. According to (1), the system boundary is a "line of demarcation between the system itself and its greater context (to include the operating environment). It defines what belongs to the system and what does not".

1.1.2 Definition of system engineering

The definition of system concept leads to defining system engineering, the discipline that is intended to support the development of systems from early requirements definition to implementation and the late stages of the product life-cycle, combining views and needs of different disciplines (55). In the literature, several definitions of the discipline of SE were proposed, we have relied on the one proposed by the INCOSE (1), which we consider the more complete.

Systems engineering is an "interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal. Systems engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user. needs" (1).

Systems engineering continues to evolve for the purpose of managing the increasing system complexity. Because system consists of interacting parts, called also subsystems (55), the complexity of subsystems can grow to the point where such subsystems may not be atomic, but be viewed as independent systems on their own merit (37) (See Figure 1.2, and can be decomposed into further subsystems. The hierarchy within a system is therefore an organizational description of the structure of the system using a partitioning relation (1).

1.1.3 From system engineering to system of systems engineering

As addressed by (2), the more and more complex customer requirements and the worldwide competition for market shares characterize increasingly product development processes. Therefore, systems engineering was in need of responding to an ever growing and diverse requirements. To address stakeholder desires, systems engineers proposed to build systems by interconnecting existing systems (37). But by coupling

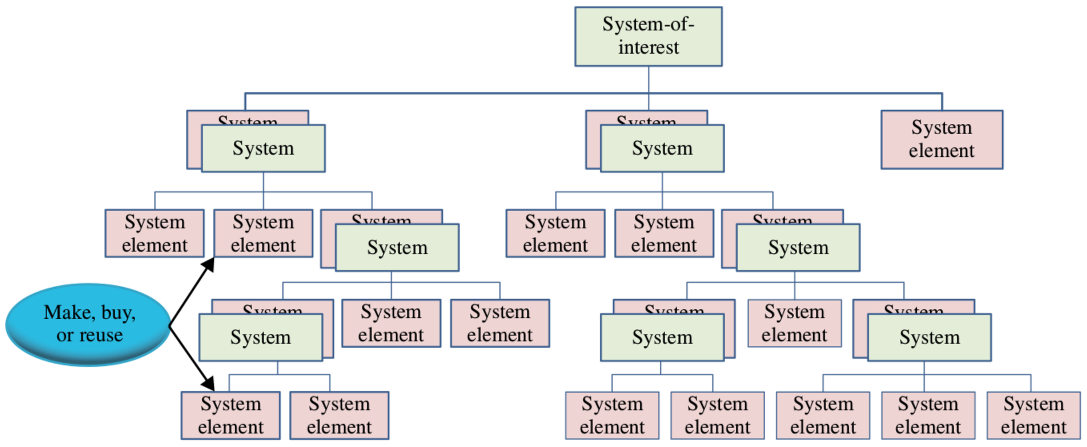


Figure 1.2: Hierarchy within a system (1)

systems, both of necessary and unnecessary complexity raises from the system designs. Complexity must be distinguished from complicated systems (56). Indeed, the degree of predominant uncertainty determines the difference between complex and complicated concerns, complex systems behavior is unpredictable because of the interconnectivity. According to (57), "the systems become complex when its parts are connected in a nonregular way. The information about the nature of interconnections is usually insufficient and this makes modeling difficult". Researchers used the term "Systems of Systems" (SoSs) to describe such complex systems; we quote as example the work of Maier (37) in which the global satellite networks is described as an SoS.

As shown in Figure 1.3, the more interconnected the system is, the more complex it is. Furthermore, SoS is a class of systems where complexity is accrued, their existence led SE discipline to emerge as an effective way to manage complexity and change. The system of systems engineering (SoSE) appeared to bring the necessary extension and evolution for SE. Unlike systems, SoS could not be described in terms of hierarchies, they are mostly represented as general networks as depicted in Figure 1.4 (3). The next section deals with SoS.

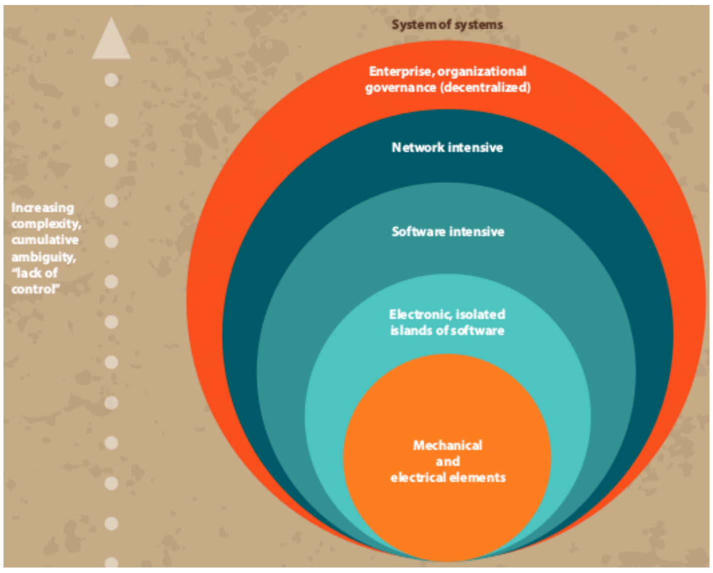

System of systems
Enterprise, organizational governance (decentralized)
Network intensive
Software intensive
Electronic, isolated islands of software
Mechanical and electrical elements
Increasing complexity, cumulative ambiguity, "lack of control"

Figure 1.3: Growing levels of systems complexity (2)

1.2 Basic concepts of SoS

As SoS field is yet an emerging area, there does not exist unifying definition of SoS, making it difficult to precisely bound the field. There are many efforts to define, characterize and categorize SoS in the literature. These different contributions are presented in the following sub-sections, beginning with an historical overview of SoSs, followed by the different definitions and key characteristics of SoSs. Next, the different categories of SoSs are presented, as well as the topology and dimensions. SoSE domain definitions and challenges are listed at the end of the section, with a focus on literature that identifies the main differences between the engineering of systems and SoSs in several aspects.

1.2.1 Brief history of SoS

Authors of (4, 5) have been interested on typical research that raised SoS. They point out that the initial work of the SoS in public journal can be dated back to 1956, when Boulding (58) envisioned SoS as "gestalt" in theoretical construction forming a "spectrum of theories" greater than the sum of its parts. After that come the works of Ack-

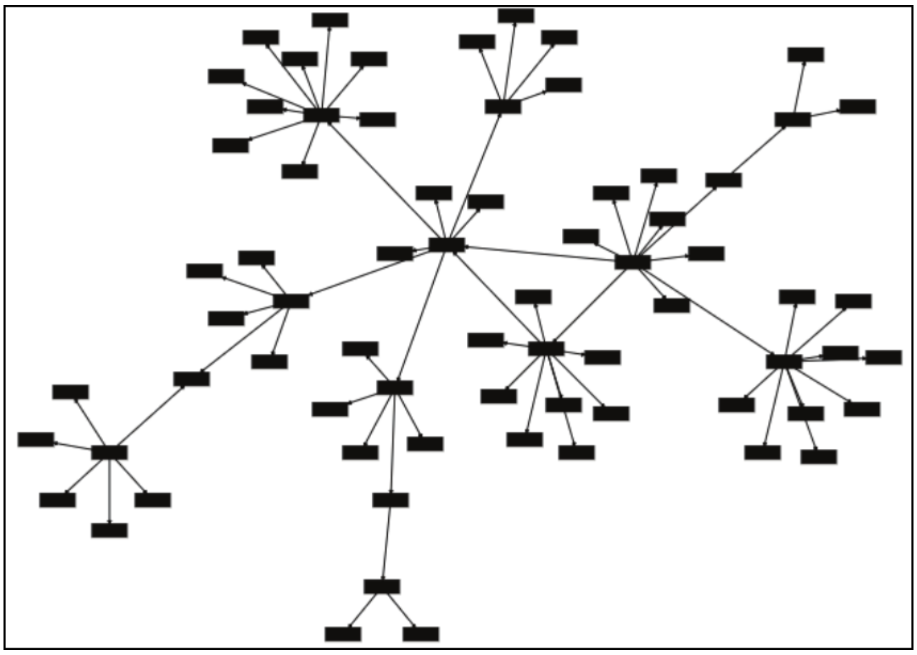


Figure 1.4: SoS described as general networks (3)

off (59), Jackson and Keys (60), and Jacob (61). Ackoff (59) in 1971 perceived SoS as a “unified or integrated set of systems concepts” and suggested that “the systems approach to problems focuses on systems taken as a whole, not on their parts taken separately”. In 1974, Jacob (61) stated that a SoS is “every object that biology studies”. Later in 1984, Jackson and Keys (60) proposed the use of “SoS methodologies” as interrelationship between various systems-based problem-solving methodologies in the field of operation research. The first use of the term “system of systems” by the Strategic Defense Initiative (SDI) [1], was used to describe an engineered technology system (62), and it wasn’t until 1989. Figure 1.5 depicts graphical time-line of the typical contributors of SoS from 1990 to 2014. It shows that industry and government applications in SoS was much later than academic research. From 2005 to the present, many organizations, government agencies, and universities have specialized in SoS research. We cite actually the example of the Office of the Secretary of Defense (OSD)

[1]anti-ballistic missile program that was designed to shoot down nuclear missiles in space

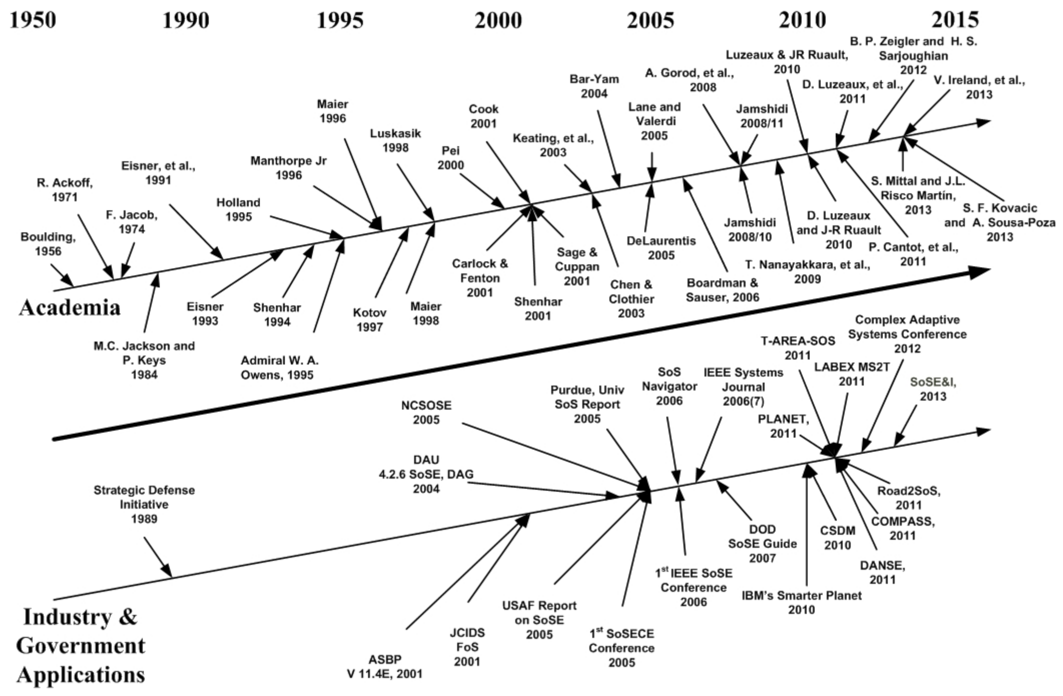


Figure 1.5: SoS Historical time-line (4) updated from (5)

[1], the SoS laboratory, part of the Center for Integrated Systems in Aerospace (CISA) of Purdue University [2], National Centers for System of Systems Engineering (NCSOSE) in Old Dominion University [3]. There is significant evidence of SoS research in several forums, we quote as example the IEEE Conference on System of Systems Engineering (IEEE SoSE) [4], the IEEE Systems Engineering Conference (IEEE SYSCON) [5], the International Council on Systems Engineering (INCOSE)[6], which manages an industry-led working group on SoS engineering. In addition, numerous journals are contributing to the advancement of studies of SoSs, such as the International journal

[1]the principal U.S. defense policy maker and adviser: https://www.defense.gov/Our-Story/Office-of-the-Secretary-of-Defense/

[2]https://engineering.purdue.edu/SoSL

[3]https://www.odu.edu/ncsose

[4]For example: http://conf.uni-obuda.hu/sose2020/

[5]http://www.ieeesyscon.org/

[6]http://www.incose.org

of System of Systems Engineering (IJSSE) [1], the IEEE Systems Journal (ISJ) [2] and the Systems Engineering journal (journal of the International Council) [3]. The most important and influential SoS research to date are presented progressively along the following subsections, whether in the academic, military or industrial field.

1.2.2 SoSs definition and characteristics

There was no widely accepted SoS definition, several researchers gave various definitions, and others preferred to distinguish SoSs by listing their characteristics. Some common definitions and characteristics are described below.

1.2.2.1 Overview of SoSs definition

The accepted modern term of SoS was arisen by the works of Eisner,et al. (63) and Shenhar (64)(See Figure 1.5). Eisner,et al., defined SoS as a “set of several independently acquired systems, each under a nominal systems engineering process; these systems are interdependent and form in their combined operations a multi-functional solution to an overall coherent mission. The optimization of each system does not guarantee the optimization of the overall system of systems” (63). This definition argues that it is important to understand the overall SoS needs, as well as to focus on balancing the interests of competing constituent systems (CSs), rather than optimizing the entire system by optimizing its components.

Shenhar’s SoS definition is “A large widespread collection or network of systems functioning together to achieve a common purpose” (64). Later, Shenhar and Bonen suggested later a two-dimensional taxonomy for systems; a technological uncertainty dimension representing the maturity level of the technologies and the system scope dimension, which is based on the complexity of the system, and categorized systems from a single unit assembly to an array of geographically dispersed systems interacting to achieve a common purpose (65). They used the term “array” as equivalent to system of systems.

[1]https://www.inderscience.com/jhome.php?jcode=ijsse
[2]https://ieeesystemsjournal.org/
[3]https://onlinelibrary.wiley.com/journal/15206858

In the same period, numerous contributions attempt to define SoS, we quote as examples the work of Holland (66), Koza (67) and Kotov (68), which was one of the first scientists to model and synthesize SoS, we did not list all the definitions because we do not aim to be exhaustive. In his book, Jamshidi (35) collected different definitions of SoS and attempt to achieve some convergence, the definition he proposed is:"systems of systems are large-scale integrated systems which are heterogeneous and independently operable on their own, but are networked together for a common goal. The goal, may be cost, performance, robustness, etc." (35). The author highlights that "each element of an SoS achieves well-substantiated goals even if they are detached from the rest of the SoS".

1.2.2.2 Definition of SoSs via key characteristics

In a response to a growing acceptance of SoSs associated to the lack of a common consensus on an SoS definition, Maier (37) identified five principal characteristics to distinguish between systems of systems and complex monolithic systems, often known by the acronym "OMGEE":

- *Operational independence* of the constituent systems, which means that the constituent systems must be able to achieve their own mission if the SoS is disassembled.
- *Managerial independence* of the constituent systems, which means that the constituent systems must be managed individually, they are acquired individually, and have their own life cycle and organization.
- *Geographical distribution* of the constituent systems means that the constituent systems are distributed over a large geographic extent. This geographical expansion is relative and relies on the available communication means and technologies. The constituent systems can exchange information and not considerable quantities of mass or energy.
- *Evolutionary development* of the SoS, which means that the SoS's objectives can change constantly and its development can be gradual. Over time, some features can be removed, modified or added, likewise, some constituent systems may be disassembled from the SoS.

- *Emergent Behaviors*, which means that the SoS capabilities could not be achieved by any of its constituent systems. Therefore, these emergent behaviors are unpredictable which lead to difficulties in validating the SoS.

Even if this definition is the most widespread, Luzeaux (69) stressed that it has some inconvenience. First, except for the possible emergent behavior, it does not consider what the SoS offers in relation to its constituent systems. Secondly, it is limited only to the technical criteria of geographical distribution and emergent behaviors but does not address the organization of technical and human resources during use. Finally, the emergent behavior and the geographical distribution of the constituent systems are not really discriminating in differentiating a system of systems from a simple set of systems, since it is complicated to identify, trace and manage emergent behavior in practice, and the geographical distribution is becoming common for many systems with the advancement of information and communication technology (ICT). It should be highlighted that in more recent work, Maier (37) argues that operational and managerial independence are the most important features in an SoS. This leads Luzeaux (69) to propose a more general definition, which a priori encompasses all the variants contained in literature: “An SoS is an assembly of systems which can potentially be acquired and/or used independently, and whose global value chain the designer, buyer and/or user is looking to maximize, at a given time and for a set of conceivable assemblies”. In this definition, Luzeaux (69) used the concept of “value chain”, that he deemed sufficient, to provide all the necessary variability to understand the difference between a system and SoS and to consider the abstraction as opposed to a simple technical vision. The author defined a value chain as “set of interdependent activities whose pursuit creates identified and, if possible, measurable value.......The value chain’s efficiency essentially relies on the coordination of the various agents involved, and their ability to form a coherent, collaborative and interdependent network.”

For Boardman and Sauser (38), both of system and SoS consists of parts, relationships and a whole that is greater than the sum of the parts, so, they are the same in that aspect. However, the two concepts vary from the manner in which parts and relationships are assembled (the composition) and thus in the nature of the emergent whole. In this direction, Boardman and Sauser (38) proposed distinguishing characteristics, that were used later by the authors to model SoS (70) and then to simulate it (71). These characteristics are as follows and could be designated by the acronym "ABCDE":

- *Autonomy*, which means that each constituent system is able to complete his own goals and without any entity's control.
- *Belonging*, that represents the ability of a constituent system to collaborate with other constituent systems within the SoS to meet a common higher purpose. The contribution of a constituent system has to be judged sufficient at the SoS level.
- *Connectivity*, which implies the capability to form connections by the dynamic distributed network as needed for the profit of the SoS.
- *Diversity*, meaning that an SoS have to be extremely diverse in term of capabilities of constituent systems. This property will make it open for evolution and adaptation.
- *Emerging*, meaning that an emergent capability results from the interactions between the constituent systems and can be attributed to the overall SoS. Boardman and Sauser argue that this property requires compliance with the other factors: "preservation of constituent systems autonomy, choosing to belong, enriched connectivity, and commitment to diversity of SoS manifestations and behavior" (38).

Abbott (39) held on the idea that SoSs are "qualitatively and structurally different from systems". He claimed that a system of systems can be understood as an environment along with the constituent systems operate and interact within it and not as a hierarchy of components. The characteristics he proposed for such environment are:

- *Open at the top*, which signifies that there is no top-level system defining the SoS, an SoS allows for the continuous introduction of new constituent systems.
- *Open at the bottom*, this ensures that at any time, the lowest SoS level, such as specific communication stack, can be modified.

- *Continually evolving, but slowly enough to be stable*, meaning that given the changes that occur in the SoS environment, an SoS is constantly evolving and is never complete. Systems of systems evolve in three different forms at least: (a) Technology changes, for example wireless replaces wires. (b) Usage changes, addition or modification of features. (c) The change of interfaces and standards. Abbott (39) specifies that most changes to a system of systems will be slow, but in the case where several small changes occur at the same time, the SoS will be conducted to "a phase change and to punctuated equilibrium effects".

1.2.2.3 Definition of SoSs in professional handbooks

In the Defense Acquisition Guidebook (72), an SoS is defined as "a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities. Both individual systems and SoS conform to the accepted definition of a system in that each consists of parts, relationships, and a whole that is greater than the sum of the parts; however, although an SoS is a system, not all systems are SoS". Note that this definition was initially stated in the (73), and then used in the Systems Engineering Guide for Systems of Systems (10), it highlighted the fact that SoS acquires new capabilities as a result of systems integration.

The INCOSE Systems Engineering Handbook (1) defined an SoS as "a system of interest (SOI) whose elements are managerially and/or operationally independent systems. These interoperating and/or integrated collections of constituent systems usually produce results unachievable by the individual systems alone. Because an SoS is itself a system, the systems engineer may choose whether to address it as either a system or as an SoS, depending on which perspective is better suited to a particular problem". The Handbook indicated also that the SoS exhibits commonly complex behaviors that result from the existence of Maier's characteristics.

In its recent edition, the ISO/IEC/IEEE 21841 (74) defined SoS as a "set of systems or system elements that interact to provide a unique capability that none of the constituent systems can accomplish on its own. System elements can be necessary to facilitate interaction of the constituent systems in the system of systems". This definition was originally proposed in the ISO/IEC/IEEE 21839 (52), and was associated with the definition of the concept of *constituent system*, which is defined as an “system that forms part of a system of systems (SoS) or more. Each constituent system is a useful system by itself, having its own development, management, utilization, goals, and resources, but interacts within the SoS to provide the unique capability of the SoS”.

We note that the concept of the *capability* of constituent system appeared in all definitions, the Systems Engineering Guide for Systems of Systems (10) defined it as “the ability to achieve a desired effect under specified standards and conditions through combinations of ways and means to perform a set of tasks”, this definition was, at the basis, proposed in (75).

1.2.3 Typology of SoSs

Since an SoS is composed of a set of interconnected systems that are managerially and operationally independent, the level of complexity of this type of system is high, as shown in the Figure 1.3. The field of SoSE tried to devise different types of SoS, this characterization offers a framework for understanding SoS to cope with the challenges arising from complexity.

Further SoSs research (40, 43) consolidated by (6, 10) and then standardized by the ISO/IEC/IEEE 21841 (74) has identified four categories of SoSs. These types, illustrated in Figure 1.6, are mainly based upon the degree of authority and responsibility governing the SoS and its evolution.

1.2.3.1 Virtual

Virtual SoS is characterized by the absence of central management and common goal. In most of the time, it is ad hoc and the constituent systems are not always known. Virtual SoS was proposed by (40), the author used the Internet and all of the services that can be founded or integrated in an ad hoc way as example of virtual SoS.

1.2.3.2 Collaborative

Collaborative SoS was defined by (40). In a collaborative SoS, there is no central authority, the constituent system engineering teams inter-operate more or less voluntarily to achieve the main common goals. (6) gave the example of the regional area disaster

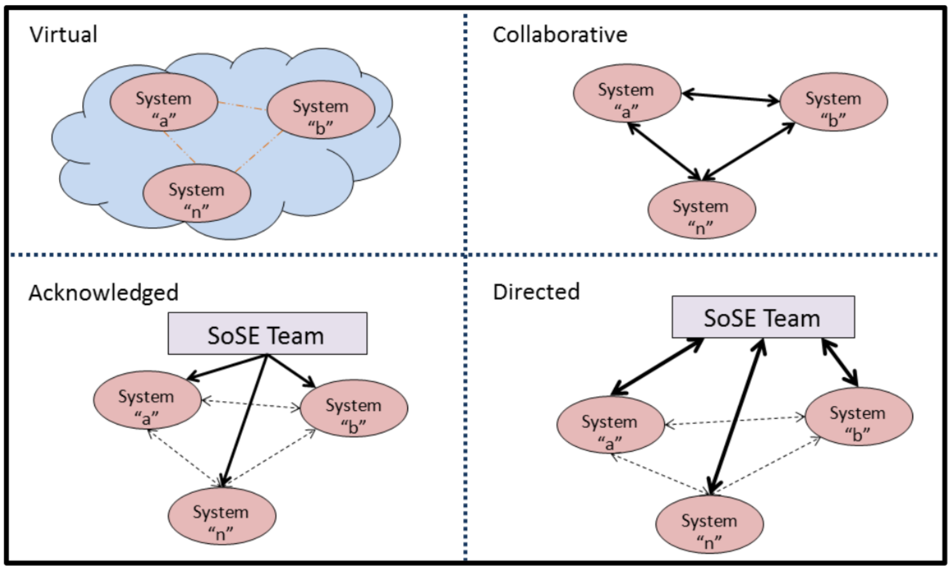


Figure 1.6: SoS Categories (6)

management system as collaborative SoS, where each organization that engages in first responder types of situations is responsible for its own systems.

1.2.3.3 Acknowledged

Proposed by (43), the acknowledged SoS has defined goals, a designated manager and resources for the SoS. It is under the responsibility of an organization, a system of systems engineering (SoSE) team that sustains the engineering of the SoS. Nevertheless, the constituent systems preserve their independent development and goals, the SoSE team does not have complete authority over them and changes in the systems rely on collaboration between the system team and the SoS team. As has been addressed by (6), the unidirectional arrows between the SoSE team and the constituent systems illustrated in Figure 1.6 signify that the SoSE team can provide directions to the constituent systems but the latter are not obliged to comply with SoSE team demands. The military command and control SoS is now considered as example of acknowledged SoS (6, 10), that has transitioned from a collaborative SoS because of the weightiness of the missions supported by the SoS.

1.2.3.4 Directed

Directed SoS (40) is developed and supervised to meet special purposes. It is centrally controlled during long-term service to continue performing those goals as well as any additional ones the system owners may wish to tackle. Constituent systems can operate independently but are controlled to fulfill the SoS goals. In Figure 1.6, the bi-directional arrows between the SoSE team and key constituent systems (but not necessarily all) means that the SoSE team has the authority to require these constituent-systems (often through some sort of contract) to develop and support SoS capabilities (6). We cite as example of directed SoS an integrated air defense network, that is usually centrally managed to defend a region against enemy systems, although its constituent systems may operate independently (43).

1.2.4 Dimensions of SoS

In response to the wide variety of SoS definitions and to the need to analyze, and evaluate techniques and methods which aim to develop and maintain SoS, Nielson and al. (36) proposed SoS dimensions. SoS dimensions were deduced from the common concepts which influence modeling and analysis that were found in SoS definitions and descriptions. The SoS dimensions were used to describe the SoS's space and could be used as base to suggest SoS modeling patterns or to analyze modeling approaches. The eight dimensions proposed by (36) are described as follows:

- Autonomy of constituents: autonomy refers to the fact that the behavior of a CS is self-controlled rather than being controlled by external rules. The degree of autonomy required by a CS may change due to the heterogeneity of an SoS, thus, modeling and analysis techniques must allow the representation of a variety of behaviors that may be performed by a CS, but that cannot be expected precisely at the SoS level.

- Independence of constituent: independence of CS implies the ability of this one to operate when disassembled from the SoS. It means that some CS behaviors are independent of the SoS, while others are associated with its role in the SoS. Thus, some CS functionalities could be hidden at SoS level and model-based techniques should consequently be able to support the hiding of information.

- Distribution: distribution is the fact that CS have to be somehow connected to be able to communicate or to share information, in general it means that CS are scattered. To support distribution, modeling frameworks need to have the possibility to connect CS using communication medium, they need also to be able to describe communication, concurrency, and to manage communication media failures.

- Evolution: evolution is the ability of an SoS to manage different nature of changes, whether in the SoS main functionality and its quality or in the composition of CS. Thus, model-based techniques in SoSE must be able to preserve the specified properties when SoS evolves. The update of CS is considered as evolution and requires conformance verification.

- Dynamic reconfiguration: dynamic reconfiguration is the aptitude of an SoS to make structural and compositional modifications, usually without expected intervention. This dimension implies the ability of an SoS to modify its configuration during service contrary to evolution, which consists to support planned updates through intervention on a gradual scale.

- Emergence of behavior: dynamic reconfiguration is the aptitude of an SoS to make structural and compositional modifications, usually without expecting intervention. This dimension implies the ability of an SoS to modify its configuration during service contrary to evolution, which consists to support planned updates through intervention on a gradual scale. SoSE modeling techniques may have abstractions for the variability of architectures and interfaces to ensure the dynamic reconfiguration

- Interdependence: interdependence corresponds to the relationship that results from the interaction of two or more CS when achieving the common SoS level goal. When CS rely on each other, it can happen that some individual behavior of a CS is sacrificed to achieve the SoS goal. Model-based approaches should provide a mean for the clear tracing of dependencies, the later could be used to determine the effect of changes of a CS. (36) pointed out that SoS engineers have to find best compromise between the degree of independence in the constituent systems and the interdependence required to reach the common goal as proposed in (76, 77).

- Interoperability: interoperability corresponds to the SoS capacity to integrate a variety of heterogeneous CS. This may require interface adaptation, standards, and protocols. Modeling and analysis techniques are facing multiple requirements to deal with interoperability. This includes strategies to verify the compatibility of CS interfaces as well as to incorporate CS heterogeneous models.

1.2.5 System of Systems Engineering definition, issues and perspectives

To address the emerging generation of complex systems which includes SoS, practitioners admitted the need to evolve the practice of SE. SoSE represents a necessary extension and evolution of traditional systems engineering discipline to engineer SoS and to cope with the different new challenges. Keating and al. (78) defined SoSE as "design, deployment, operation and transformation of meta-systems, that must function as an integrated complex system to produce desirable results". Several surveys sustained that at this point of SoSE development, there is no SoSE precise and unified definition (7), the Systems Engineering Guide for Systems of Systems (10) suggested that SoSE deals with "planning, analyzing, organizing, and integrating the capabilities of a mix of existing and new systems into an SoS capability greater than the sum of the capabilities of the constituent parts". The guide used the term SoS SE (System of Systems System Engineering) to describe SoSE and considers that SoSE addresses SE considerations for SoS.

The INCOSE Systems Engineering Handbook (1) does not define SoSE, it considers that SE is a domain that deals with all kinds of systems, even if the processes and methods used for each kind are different. Furthermore, the handbook assumes that "SoS is itself a system, and the systems engineer may choose whether to address it as either a system or as an SoS, depending on which perspective is better suited to a particular problem". In this manuscript, we assume that SoSE is a sub discipline of SE that deals with SoS.

According to (36, 76), SoSE's principal research areas could be classified into the following issues:

- Modeling and architecting: the development of models in which there is the use of existing systems as components of the SoS, and optimize the architecture while taking into account the SoS dimensions. Furthermore, the use of Domain-Specific Languages (DSL) to analyze SoS mission and capability objectives, and to set the concepts of operational development.
- Simulation: the proposition of simulation tools to analyze and understand the complexity of SoS behavior.
- Testing: the implementation of testing techniques in case of: complex and large SoS, the use of different standards, multi-stakeholder situation, dynamic evolution of SoS configurations,.etc.
- Verification: the development of verification tools to support simulation and testing, for analyzing different properties.

In the literature review presented by Keating and Katina (8), the authors claimed that the SoSE issues tend to evolve among three perspectives: military, academic, and enterprise. Figure 1.7 shows the three diverging perspectives of SoSE.

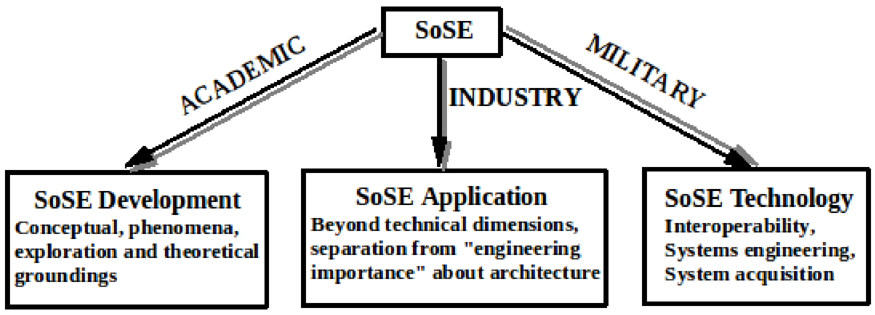

SoSE
ACADEMIC
INDUSTRY
MILITARY
SoSE Development
Conceptual, phenomena, exploration and theoretical groundings
SoSE Application
Beyond technical dimensions, separation from "engineering importance" about architecture
SoSE Technology
Interoperability, Systems engineering, System acquisition

Figure 1.7: The SoSE perspectives (7) adapted from (8)

The first perspective is the military perspective, mainly motivated by an emerging viewpoint within the US DoD. This viewpoint consists of designing technical command and control systems to operate independently while being highly interoperable. Such systems require the incorporation of different technology systems to realize the SoS mission. The second emerging perspective is the academic perspective, It provides a specific and more extensive look at the nature and development of the SoSE field. It shows the direction for more rigorous development along philosophical and theoretical aspects. The third emerging perspective is the enterprise perspective. This perspective encompasses a more holistic view of SoSE, considering the notion of the enterprise as a system of systems that exists outside the purely technical view, and which is dominated by architecture. As argued by (8), each perspective carries a logic that offers its own validity, to the community which develops the perspective. However, the intersection of the multiple perspectives have to be examined, to avoid any source of divergence in the SoSE field, that which may hinder the coherent development of the field.

1.2.6 Difference between SE and SoSE (SoSE key challenges)

As mentioned by (33), both of the characteristics of an SoS as presented in section 1.2.2.2, and the relationships of authority between the constituent systems and the SoS as described in the categories of SoS as presented in section 1.2.3, influence how system engineering can be applied to SoS. Table 1.1 lists the main differences between the engineering of systems or SoS in four aspects (10, 33): management and oversight, operational focus, implementation and engineering and design considerations. The rest of this subsection deals with the main environmental distinctions.

- Management and oversight: the first aspect that affects the application of SE to SoS is the community in which SoS is built. As described in Table 1.1, there are many key variations. On one hand, single systems have their own owners, objectives, resources, and funding. The engineering focus is on the physical system design and implementation, stakeholders are committed to that system and play specific roles in the SE of that system. On the other hand, SoS management must consider CS constraints. SoS stakeholders may have conflicting interests with CS stakeholders. These latter may give low priority for SoS needs. The governance of an SoS is difficult since it can involve the collection of agencies, authority structures, and the coordination required for allocating resources and organizing activities.

- Operational focus (goals): the objectives within a single system are well-defined. In the case of SoS, new needs on the CS may appear following CS assembly, for functionality or information sharing, which had not been considered in their individual designs. Furthermore, SoS objectives may not be aligned with those of the CS. Very often, these systems have to support both their own objectives and new SoS objectives.

- Implementation: in the case of a single system, the engineering is done using a well-established process and activities with clear decision points. Testing and evaluation of the entire system, or at least the subsystems related to specified requirements are generally possible. As long as SoS implies several legacy CS, new CS, developmental CS, technology insertion, or life extension programs, which are not necessarily at the same phase of development, the SoSE must consider and leverage the asynchronous development life cycles of the individual systems. Because of all these challenges, it is often very difficult to completely test and evaluate SoS capabilities.

- Engineering and design considerations: Engineering of single systems implies 1) the definition of system boundaries, which is often a static problem that consists of separating what is inside the system from what is outside, as well as 2) the definition of interfaces requirements, and 3) developing approaches to ensure the performance and behavior. In contrast, SoS boundary is often ambiguous. It depends on the capabilities needed so that the SoS can meet his objectives. The needed capabilities determine the CS expected to be involved in the SoS. Moreover, the performance of an SoS is affected by the CS performance, and the end-to-end behavior of the ensemble of systems, that could determine the key issues which affect that behavior.

Table 1.1: Comparing SE and SoSE (Adapted from (5, 10, 33))

Table 1.1: Comparing SE and SoSE (Adapted from (5, 10, 33))

Aspects of Environment	**SE**	**SoSE**
Management and Oversight		
System	Physical engineering	Socio-technical management and engineering

Stakeholder involvement	Clear stakeholder set	Two levels of stakeholders: CS level and SoS level stakeholders. Their priorities are different and interests may be conflicting. Not all CS stakeholders are recognized at SoS level
Owners	Clear ownership with the possibility to transfer resources among elements	Multiple owners taking individual decisions concerning resources
Governance	Aligned management and funding	Management and funding at SoS level and CS level, which make them more complex because the SoS does not necessarily have authority over all systems
Operational Focus (Goals)		
Operational focus	Designed and developed to achieve clear set of operational objectives	Set of operational objectives that have to be achieved using CS whose objectives may not necessarily align with the SoS objectives
Implementation		
Acquisition/ Development	Single system life cycle aligned to establish acquisition and development processes	Continuous SoS lifecycle intertwined with multiple system lifecycles across asynchronous acquisition and development efforts involving legacy systems, systems under development, new developments, and technology insertion
Process	Well-defined process	Continuous process adaptation after learning
Test and Evaluation	Usually, it is possible to test and evaluate the system	The difficulty of synchronization of several CS' life cycles makes the test and evaluation more complex
Engineering and Design Considerations		
Boundaries and Interfaces	Based on the identification of boundaries and interfaces for the single system	Based on the identification of possible and interoperable CS that will contribute to meet SoS objectives, while balancing their own objectives
performance and Behavior	The performance of the single system to accomplish his objectives	The performance of the SoS that allows it to meet his objectives while considering the CS own objectives

Metrics	Well-determined (The INCOSE Handbook for example)	Difficult to defined, agree, and quantify

1.3 Discussion

Even if there does not exist unifying definition of SoS, several features and artifacts have been cited throughout the chapter. These features and artifacts were proposed to characterize SoS from traditional systems. It then becomes clear that a SoS is qualitatively and structurally different from a traditional system. Both of the characteristics of an SoS, and authority relationships between the CSs and the SoS influence on how to engineer SoSs. Furthermore, they influence how system engineering (SE) can be applied to SoSs in several aspects like management and oversight, operational focus, engineering and design considerations (10**?**). SoSE is primarily concerned with acknowledged SoS, where the SoS is under the responsibility of an organization. As there is a difference in engineering systems and SoSs, practitioners admitted the need to develop methods, processes and tools to analyze, design, architect, and evolve SoS. Next chapter provides a state of the art on the different approaches and principles of SoSs modeling.

Chapter 2

SYSTEM OF SYSTEMS MODELING: STATE OF THE ART

In the development of monolithic systems, a system engineering process is used to facilitate the development and implementation of the system. It begins with the requirements elicitation and finishes with the validation of the solution. However, when engineering SoSs, specific characteristics have to be considered as addressed in the previous chapter, such as the independence of the CSs. Modeling plays a prominent role in the development of SoSs. It supports the analysis of the SoS requirements and the design of the architecture of the SoS, as well as its verification. This chapter presents an overview of the state of the art on SoSs modeling. It starts by presenting SoSE life cycle models, and SoSE architecting artifacts. It lists, then the different SoSs modeling contributions that are most closely related to our work. The chapter ends with a discussion of the state of the art and addresses a research gap analysis.

2.1 Systems of systems development phases

Given the distinctions between classical SE and SoSE due to the additional challenges that raised with SoS paradigm. SoSE phases used to develop SoS could be considered as an essential extension and evolution of classic SE. In fact, systems engineers when applying SE to SosS adapted the SE life cycle, activities and processes to SoS. In the following, we describe the SoSE life cycles proposed in the literature as well as the SoS architecting artifacts.

2.1.1 Systems of systems development life cycles

This section presents the most well known SoSE life cycles that were proposed in response to several SoSE key challenges. These life cycles were needed to consider the fact that SoS is composed of new, modified, or existing systems, and that certain systems may evolve and their future is uncertain.

2.1.1.1 SoSE Double V Model

By considering the constituent systems involvements in SoS, the SoS development process was represented at the beginning by "two-tiered development in a double V model" (9) as depicted in Figure 2.1. The engineering of constituent systems is done in parallel with the engineering of the SoS and SoS evolution relies on constituent systems changes through their own life cycles.

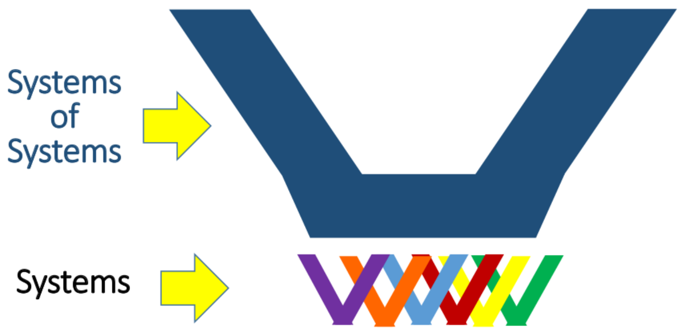


Figure 2.1: The SoSE Double V Model (9)

Even if the double V model takes into account the existence of constituent systems as recommended for SoSE phases in the ISO/IEC/IEEE 21839 (52), it doesn't consider other characteristics of SoS as the evolutionary development and the dynamicity and changes in the constituent systems because it implies "a single pass development of a SoS" (9). Furthermore, some reviews have demonstrated that the classical system engineering process, including need statement, requirements, implementation, validation and delivery, and eventual disposal is not suitable for SoS given their evolutionary character (10). The double V model was presented also by Clark (79) and was designated by the name SoSE Dual-V Model.

2.1.1.2 SoSE trapeze model

The SoSE trapeze model (10, 80, 81) was proposed after a review done by the Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics (OUSD AT&L) that concerned defense programs, where SE had to be applied in an SoS context, and which has shown that a SoSE process is needed to offer guidance for the DoD SE community. The SoSE trapeze model is depicted in Figure 2.2, it is especially applicable to acknowledged SoS type, it is based on the necessity to understand the SoS objectives, to analyze the constituent system's capabilities, and to consider the evolutionary character of SoS. Seven core elements characterize the trapeze model and

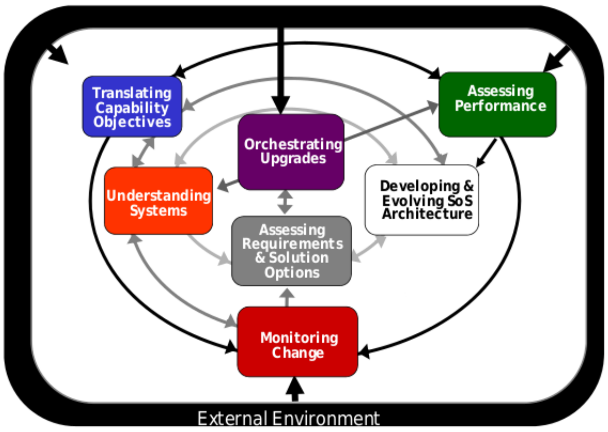

Translating Capability Objectives
Assessing Performance
Orchestrating Upgrades
Understanding Systems
Developing & Evolving SoS Architecture
Assessing Requirements & Solution Options
Monitoring Change
External Environment

Figure 2.2: The SoSE Trapeze Model (10)

are described as follows (10, 80, 81): 1) "translating the SoS capability objectives into requirements" and 2) "assessing the performance pertaining to these capability objectives" as well as 3) "monitoring and assessing the external changes on the SoS". It is important for SoSE engineers to 4) "understand systems that contribute to the realization of SoS objectives and their relationships" and 5) "to develop and evolve an SoS architecture". For the SoSE, it is crucial to 6) "address new requirements and solution options" and 7) "orchestrate upgrades to the SoS and implement the changes".

2.1.1.3 Wave model

Dahmann (11) "built on the trapeze model and translated the SoSE core elements, their interrelationships, and SoS decision making artifacts to a more familiar and intuitive wave model representation". This model views SoS development as an evolutionary, iterative, and incremental process to adapt constituent systems and improve SoS performance. Originally, the wave planning was introduced by Dombkins (82), and it was subsequently "applied to the SoS trapeze model to illustrate the incremental and iterative process that characterizes acknowledged SoS development" (11). The SoSE wave model is illustrated in Figure 2.3. Six steps characterize the SoSE wave model (11).

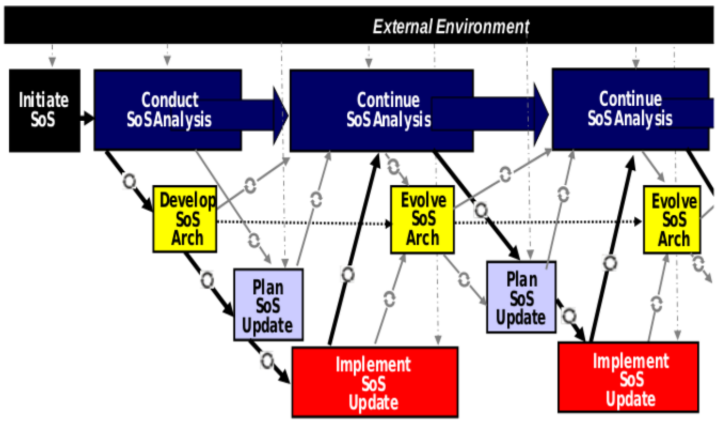


Figure 2.3: SoSE Wave Model (11)

In the SoSE wave model, systems engineers are actors in 1) initiating the SoS by understanding SoS goals and 2) conducting SoS analysis by taking into account several artifacts such as SoS performance measures and SoS risks and mitigations. Fundamental to SoSE is the 3) development and evaluation of the SoS architecture as well as the 4) planning of SoS updates and evaluation of the SoS priorities. The SoSE team is involved in 5) monitoring the implementations at the constituent system level and finally 6) performing a continual SoS analysis to revisit key information.

2.1.1.4 SoSE DANSE model

Another way to represent the SoSE life cycle is the one proposed by the European commission project Designing for Adaptability and EvolutioN in System of systems Engineering (DANSE) [1]. Like the wave model, the SoSE DANSE model (9, 12, 83) relies on evolutionary process as shown on Figure 2.4. The DANSE model is based

[1]European funded project. Web site: http://www.danse-ip.eu Accessed 19/06/2020

on the idea that SoS grows from interacting constituent systems rather than being designed from above. Furthermore, the DANSE model considers that changes in an SoS are caused by three factors: modifications to the constituent systems, environmental change, and changes made by an SoS Manager.

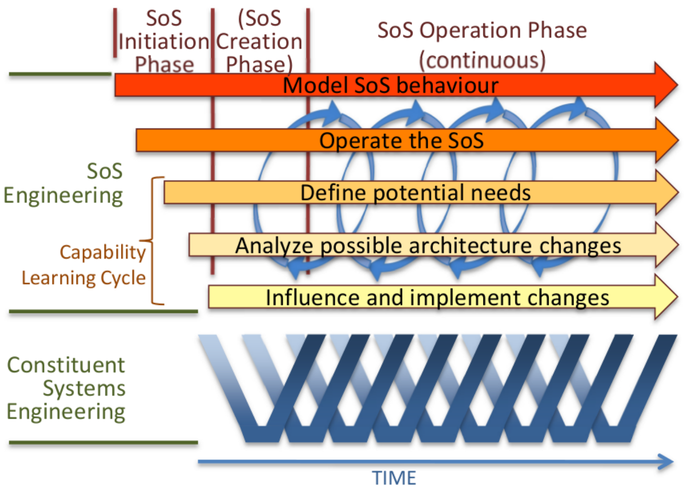

SoS Initiation Phase
(SoS Creation Phase)
SoS Operation Phase (continuous)
Model SoS behaviour
Operate the SoS
SoS Engineering
Define potential needs
Capability Learning Cycle
Analyze possible architecture changes
Influence and implement changes
Constituent Systems Engineering
TIME

Figure 2.4: The SoSE DANSE Model (12)

Three phases characterize the DANSE life cycle, they are described at the top of the Figure 2.4 and define vertical compartments. In the 1) SoS initiation phase, initial SoS is created with newly participating CS, who their own goals match with SoS goals. contributes SoS. In the 2) SoS creation phase, the SoS manager evaluates the emergent behaviors that result from the initiation phase, design the SoS and guide the evolution of the SoS during 3) SoS operation phase.

2.1.2 Systems of systems architecting artifacts

The proposition of SoSE specific life cycles specific proves that there is already a greater awareness of SoS characteristics, which create SE challenges, and a recognition that some activities are unique to SoSE. Always for the same purpose, and to help employ fundamental SE processes in an SoS environment, Dahmann and Lane (34) define SoSE artifacts, compare them with similar ones developed and used for single systems, and discuss how they are used to guide SoS engineering processes. The artifacts are presented in Table 2.1.

Table 2.1: Comparing SoS and system artifacts (Adapted from (11, 34))

Table 2.1: Comparing SoS and system artifacts (Adapted from (11, 34))

Artifact	**System**	**SoS**
SoS capability-related information		
Capability objectives	Focuses on a user's capability gap, and could provide functionality for supporting SoS objectives	The focus is on top-level SoS capabilities. Several constituent systems, often not known initially, contribute to the SoS mission accomplishment
CONOPS	Single system focus	Based on numerous systems
Requirements	Defined by analyzing the operational users needs	SoS requirements space is bound by SoS user's needs. It must be refined until the identification of specific requirements, which will be handled by the constituent systems.
Constituent system-related information		
Systems information	The emphasis is on interfaces between parts, the exchange with external systems, and the enhancement of system performances. Focus is typically on the technical aspects	Focuses on constituent system-level information that affects the SoS objectives. Includes in addition to the technical aspects, the operational, organizational, and planning issues.
SoS technical information		
Performance measures and methods	The emphasis is on performance of the specific system and the interactions within external interfaces	The emphasis is on SoS solution performance while remaining as much as possible independent from constituent systems, to permit the assessment of other alternative solutions
Performance data	Mainly gathered in typical SE life-cycle. Used to assist fielding decisions	Predominantly gathered from different operational environments. Used to detect capability gaps and to improve continuously the SoS

Architecture	Provides top-level system components, their connections, the communication protocols, and all key elements that link components to the external system interface. Architecture is used to make decisions on system development	Includes constituent systems, their connections, the communication protocols, and all key elements that link constituent systems. Architecture is used to understand the relationships among constituent systems, and to develop solution options to meet SoS objectives
SoS technical baselines	Specific system artifacts/components which constitute the system baselines	Emphasis is on SoS-level definition and identification of constituent system baselines that comprise the SoS baseline
SoS management & planning information		
SE planning elements	Takes the form of an SE Plan, and is part of the acquisition process	The main important point is the determination of the rhythm, organizational structure, technical reviews, and decision processes across SoS evolution. It is important to consider the ability and willingness of constituent systems to support SoS
Master plan	Focus is on the defined end-state of the system. Reflects the system acquisition strategy	The emphasis is on SoS-level view through several increments, and interaction points of constituent systems. Continuous improvement is an important consideration
Agreements	Emphasis is on identifying system dependencies (e.g. need of special components)	Emphasis is on assisting relationships within multiple organizations
Technical plan(s)	Aims to incorporate system changes	The emphasis is on making modifications to constituent systems to achieve an increment of SoS development

Integrated master schedule	Full list of SE development activities, milestones and related timetables	Collection of SoSE development activities and milestones associated to consitituent systems SE activities and milestones. The emphasis is on constituent systems coordination key elements, and pointers to their schedules for the current SoS increment
Risks and mitigations	Emphasis is on system challenges and expected problems. Includes current dependencies which present specific risks	Emphasis is on desirable and undesirable emergent behaviors of the SoS. Includes also constituent systems risks

According to (11, 34), the critical information to realize effective SoSE and the corresponding artifacts could be organized as follows:

- SoS capability-related information: gives the SoSE capability context, which is defined early in the SoSE process. The SoS capability objectives correspond to the main goals of the the SoS, the SoS CONcept Of OPerationS (CONOPS) defines the use of the SoS constituent system functionality in an operational context, and the SoS requirement space bounds the operational tasks and missions while considering the environment change that affect the execution of the required functions.

- Constituent system-related information: is captured early in the process when the SoS is initiated. The systems information corresponds to information pertaining to the systems that impacts the SoS capability objectives, and this information is collected to be used for replacements as the SoS evolves.

- SoS technical information: the SoS performance measures and methods capture the basis for assessing the overall performance of the SoS and for improving the SoS, while the effectiveness data of the SoS are collected from different environments to identify the areas needing more attention. The SoS architecture determines how the constituent systems operate together and tackles their implementation only when the related functionality is needed at SoS level. Finally, the

SoS technical baselines are developed for each increment of SoS development. SoS baselines include requirements, functional, allocated, and product baselines, as well as constituent system baselines that are managed by the systems themselves.

- SoS management & planning information: include the six principal elements (See Table 2.1). The SoSE planning elements determine "the rhythm, technical reviews, and decision processes across the SoS evolution". These elements furnish also "the principal SE rules of engagement for the SoS and are utilized by all SoS actors". Master plan is an integrated plan offering a top-level view of several upgrades to implement the SoS evolution strategy. This plan is established by the SoSE team and the SE teams.. Agreements define SoS participant's roles and responsibilities, along with their contributions in development increment. Technical plans are established for each SoS update cycle and regroup SoS implementation, integration, and test plans. Integrated master schedule are created for each SoS development increment, they contain the crucial aspects in the technical plans that need to be tackled in SoS development. Finally, the SoS risks are captured and tracked.

2.2 Model Based System Engineering (MBSE) via the System Modeling Language (SysML)

Model Based System Engineering (MBSE) makes system development easier by employing a collection of computer-interpretable models (84) to describe, formalize, and organize the system development process. Models are useful at several stages of the system development process, from requirements to system verification and implementation. According to the INCOSE Systems Engineering Vision 2020 (85), MBSE is defined as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases". Several other acronyms such as Model Driven development (MDD), Model Driven Engineering (MDE) and Model Based Design (MBD) are synonymous with MBSE. These acronyms are in fact until now more often used in the software engineering community in which MBSE then stands for model-based software engineering.

Researches in SoSE domain claim that the use of MBSE to model systems is a promising path, the adoption of MBSE approach brings five benefits according to the INCOSE (86): improved communications, increased ability to manage system complexity, improved product quality, enhanced knowledge capture and reuse of information, and improved ability to teach and learn SE fundamentals. Indeed, models become then the primary means of communication between engineers, clients, builders and users. According to IEEE 610.12 (87), a model is "approximation representation, or idealization of selected aspects of the structure, behavior, operation or other characteristics of a real-world process, concept, or system". MBSE links and integrates different system modeling activities (13). It involves capturing the various aspects of an integrated system or SoS and incorporating them into a single model. A model will consist of various graphical viewpoints that describe the characteristics of a system (See Figure 2.5).

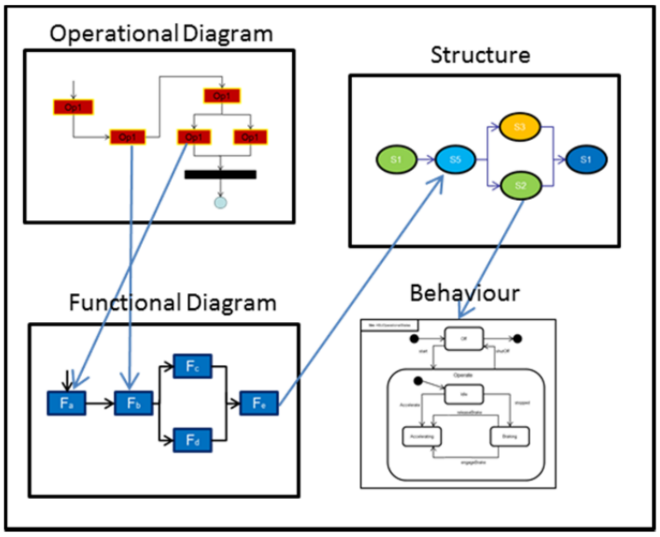


Figure 2.5: System Model Example (13)

The system models are built around the aim of the modeling and the required viewpoints. According to IEEE 1471 (88), a viewpoint is "a template, pattern or specification for constructing a view". A viewpoint usually informally specifies "the purpose of a perspective and the concerns that stakeholders wish to address" (84). Based on the required model viewpoints, the modeling language and the tools are selected. Because of its capacity to capture the structure of the constituent systems, **System Modeling**

Language (SysML) [1] is widely used and provides a strong foundation for system modeling. It is based on the Unified Modeling Language (UML) [2] but caters in both semantics and usage specifically towards systems engineering as opposed to UML's focus on software development. Many of the basic elements, interactions, and views from UML are included or extended within SysML. It is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. There are four sets of viewpoints that are captured in SysML (four pillars of SysML): structural, behavioral, requirements, and parametric. Structural viewpoints establish the definition of elements: the composition of systems, their properties, and organizational grouping. Behavioral viewpoints describe how these elements function, their operational states, and their interactions. The requirements viewpoint allows SE engineers to create, relate, trace, and analyze the system requirements. Finally, parametric viewpoints allow for the application of constraints on systems via logical and mathematical expressions.

2.3 State of the art on SoSs modeling

This section is the literature review of different approaches that tackle with the design and development of SoS. We first present methodologies proposed to develop complex systems. We then present different processes, which were proposed as part of a research project. Some related research proposed in the SoSE field are given. At the end of the section, a discussion which concentrates on advantages and disadvantages of the approaches is given.

2.3.1 Complex systems engineering methodologies

A number of SE methodologies are currently used by the systems engineering community, offering guidance for analyzing, developing and documenting complex systems.

[1]http://www.omgsysml.org/ Accessed 19/11/2021
[2]http://www.omg.org/UML Accessed 19/11/2021

2.3.1.1 OOSEM

"The Object-Oriented Systems Engineering Method (OOSEM) provides an integrated framework that combines object-oriented techniques, a model-based design approach and traditional top-down SE practices" (14). OOSEM is now advocated as an example of a model-based systems engineering (MBSE) best practice since it was realigned with SysML (it was initially based on the unified modeling language (UML). the following activities are encompassed in the OOSEM "specification and design system process" as shown in Figure 2.6: (1) analysis of the stakeholder needs, and (2) analysis of the system requirements. (3) Logical architecture definition by highlighting how the logical components interact to fulfill the requirements. The (4) allocation of hardware, software, data, and procedures to the logical components. The (5) activity of optimizing and evaluating alternatives and finally, the (6) activity of managing the traceability of requirements from the mission level requirements to the component requirements.

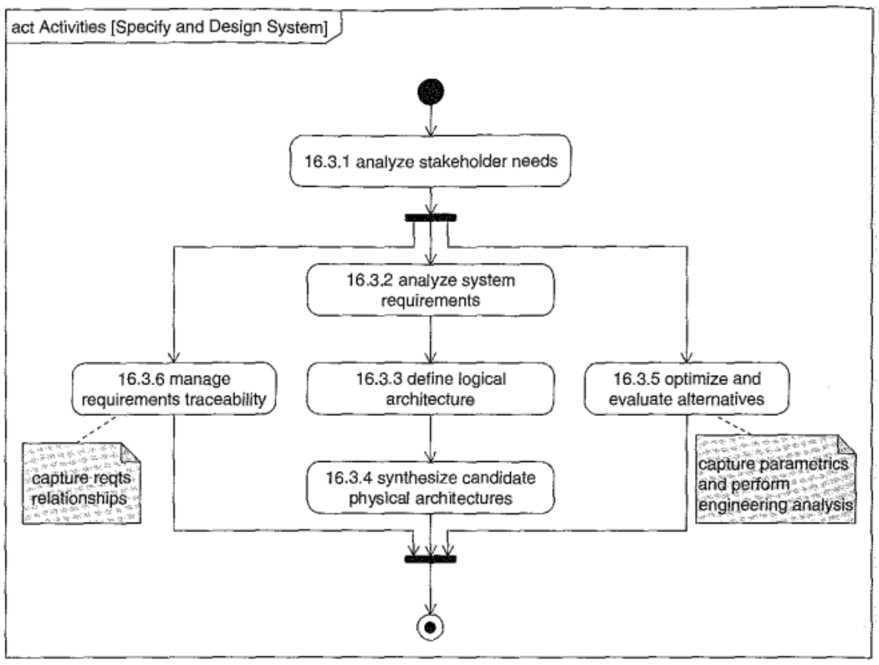

act Activities [Specify and Design System]
16.3.1 analyze stakeholder needs
16.3.2 analyze system requirements
16.3.6 manage requirements traceability
16.3.3 define logical architecture
16.3.5 optimize and evaluate alternatives
capture reqts relationships
16.3.4 synthesize candidate physical architectures
capture parametrics and perform engineering analysis

Figure 2.6: The OOSEM Specify and Design Process (14)

2.3.1.2 Harmony

The Harmony (15, 45) SE process is characterized by three main activities depicted in Figure 2.7: (1) "requirements analysis", (2) "system functional analysis", and (3) "architectural design". The Harmony SE is a model based process and uses SysML as the modeling language. In the Harmony "requirements analysis phase", requirements are grouped into use cases. The "system functional analysis" phase consists of translating functional requirements into set of system functions. A black box model is related to each use case. Incrementally, these black box models are aggregated into a black box system model. The "architectural design" activity is composed of the "system architectural design" and "subsystem architectural design" elements. In the subsequent system architectural design phase, the valid operational contracts is assigned, based on performance and safety requirements, to the physical architecture. The subsequent subsystem architectural design phase aims at deciding the operational contracts within a physical subsystem that should be implemented in the hardware and software (hardware/software tradeoff analysis).

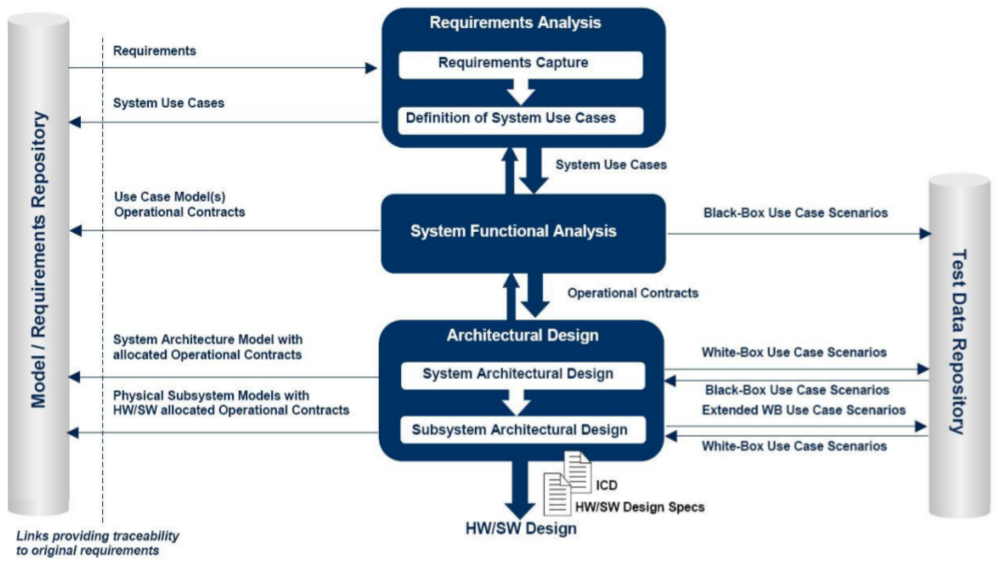
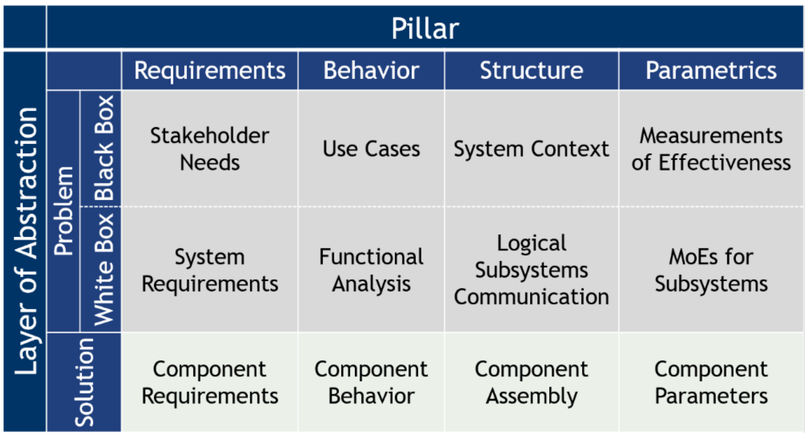

Model / Requirements Repository
Requirements
System Use Cases
Use Case Model(s)
Operational Contracts
System Architecture Model with allocated Operational Contracts
Physical Subsystem Models with HW/SW allocated Operational Contracts
Links providing traceability to original requirements
Requirements Analysis
Requirements Capture
Definition of System Use Cases
System Use Cases
System Functional Analysis
Operational Contracts
Architectural Design
System Architectural Design
Subsystem Architectural Design
ICD
HW/SW Design Specs
HW/SW Design
Black-Box Use Case Scenarios
White-Box Use Case Scenarios
Black-Box Use Case Scenarios
Extended WB Use Case Scenarios
White-Box Use Case Scenarios
Test Data Repository

Figure 2.7: The Harmony Process (15)

2.3.1.3 MagicGrid

MagicGrid (16) is a SysML-based framework for modeling complex systems. As depicted in Figure 2.8, the MagicGrid framework consists of viewpoints (black box, white box and solution) and aspects (the four pillars of SysML: requirements, system structure, system behavior, and parameters) organized in a grid view. The cells of the grid represent different views of model-based systems engineering, which are described as follows (16): (1) the requirement elicitation of stakeholders by using the SysML requirement diagram (RE); (2) a use case description of the refinements of functional stakeholder needs; (3) system context representation using the SysML internal block diagram (ibd); (4) measures of effectiveness (MoEs), which indicate the nonfunctional requirements, described in the SysML block definition diagram (BDD). The MoEs calculation procedures are specified with the SysML parametric diagrams. The (5) identification and specification of system requirements is performed by using the RE diagram, (6) and functional analysis elaboration is performed with multiple SysML activity diagrams, specifying internal system functions. The (7) logical subsystem communication identification is established using the control and resource flows defined in the functional analysis model. Both of the SysML BDD and SysML IBD are used to capture this view. The (8) measures of effectiveness (MoEs) as well as the measures of performance (MoPs) are captured for each logical subsystem, in the SysML BDDand parametric diagrams. The (9) component requirements are captured using the SysML requirement diagram. The (10) component behavior definition is performed using an association of SysML state machine, activity, and sequence diagrams; (11) component structure elaboration is performed by illustrating the physical connections between physical components, and this view is captured using both the SysML BDD and IBD. The (12) component parameter definition of each component is performed, in which each parameter captures the component characteristics and the links between them and describes how the MoEs and MoPs already specified are accomplished using these characteristics.

Other interesting SE approaches have also been reported in the literature, such as the IBM Rational Unified Process for Systems Engineering (RUP SE) (89), JPL State Analysis (SA) (90), and SYStem MODeling (SYSMOD).



Layer of Abstraction			Pillar			
			Requirements	Behavior	Structure	Parametrics
	Problem	Black Box	Stakeholder Needs	Use Cases	System Context	Measurements of Effectiveness
		White Box	System Requirements	Functional Analysis	Logical Subsystems Communication	MoEs for Subsystems
	Solution		Component Requirements	Component Behavior	Component Assembly	Component Parameters

Figure 2.8: The MagicGrid Framework (16)

2.3.2 Enterprise architecting frameworks

Architecture frameworks provide a roadmap for describing the architecture of a system. These descriptions are necessarily done from multiple view-points, not from single viewpoint. we present the frameworks that are more suited to SoS architecture than others.

2.3.3 DoDAF

The Department of Defense Architecture Framework (DoDAF) (46) is an architecture framework for the United States Department of Defense (DoD). In the DoDAF framework, there is several views, each of which is broken down into products and data: operational view, systems and services view, etc. The operational view aims to describe the tasks and activities, operational elements, and resource flow exchanges required to conduct operations. The capability view aims to describe the mapping between the required capabilities and the activities that enable those capabilities. Managers in the DoD are required to specify the requirements and control the development of architectures for any defense systems they procure. Since it is the preferred system architecture, suppliers to the DoD should adopt DoDAF as the primary baseline for any defense systems they are contracted to build. The core of DoDAF is a data-centric approach where the creation of architectures to support decision-making is secondary to the collection, storage, and maintenance of data needed to make efficient and effective decisions. Figure 2.9 shows an overview of the conceptual framework of DoDAF.

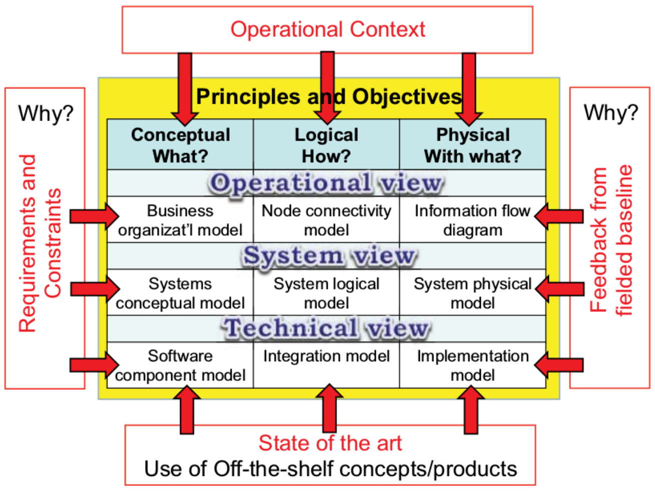


Figure 2.9: DoDAF Views and Concepts (17)

2.3.4 MoDAF

The Ministry of Defense Architecture Framework (MoDAF) (47) is an architecture framework for the UK Ministry of Defense. Similar to DoDAF, MoDAF specifies a set of rules and templates, known as views, which provide a visualization of a particular business area within an enterprise, which are aimed at the various stakeholders who interact with the enterprise. These views represent a window into the enterprise architecture that is specific to the particular stakeholder conducting the viewing. The views are divided into seven categories: **(1) strategic views**, to describe the desired business outcomes; **(2) operational views**, to define the processes, information, and entities needed to fulfill the capability requirements; **(3) service oriented views** to represent the services that are required to support the processes, **(4) systems views**, specify the physical implementation of the operational and service orientated views; **(5) acquisition views**, to define the dependencies and timelines of the projects that will deliver the solution; **(6) technical views**, to represent the standards, rules, policies, and guidance that are to be applied to the architecture; **(7) all views**, used to provide a description and glossary of the complete project architecture, including scope, ownership, timeframe, and all other meta data that is necessary to effectively query architectural models. Figure 2.10 illustrates how the different viewpoints relate to each other.

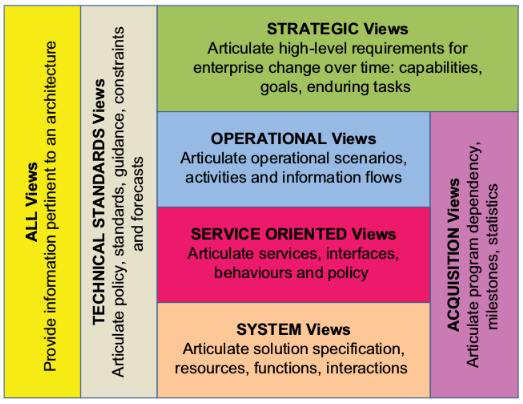

ALL Views
Provide information pertinent to an architecture
TECHNICAL STANDARDS Views
Articulate policy, standards, guidance, constraints and forecasts
STRATEGIC Views
Articulate high-level requirements for enterprise change over time: capabilities, goals, enduring tasks
OPERATIONAL Views
Articulate operational scenarios, activities and information flows
SERVICE ORIENTED Views
Articulate services, interfaces, behaviours and policy
SYSTEM Views
Articulate solution specification, resources, functions, interactions
ACQUISITION Views
Articulate program dependency, milestones, statistics

Figure 2.10: MoDAF views (17)

While DoDAF and MoDAF frameworks have similar views, their respective metamodels are different. Despite this, the MoD has been working with the Object Management Group (OMG) to develop the Unified Profile for DoDAF and MODAF (UPDM), an abstract UML profile that implements the MODAF metamodel and the DODAF metamodel. It is based on the Unified Modelling Language (UML) and extends the Systems Modeling Language (SysML). The UPDM prescribes more than 40 views. The viewpoints allow modeling in different levels of abstraction, and are rich in term of concepts, including all the concepts related to SE or SoSE domains.

2.3.5 Research projects

This section collects a few solutions that have been proposed in the context of research projects. Among all approaches, we choose the solutions proposed in COMPASS, DANSE, and AMADEOS projects.

2.3.5.1 DANSE methodology

In the 2.1.1.4 section, we presented the SoSE DANSE lifecycle, and described the main steps that characterize it. In what follows, we detail the associated methodology. The DANSE methodology is described in terms of six engineering activities (9), shown as the colored horizontal bars in the Figure 2.4. The first activity is to (1) model the SoS behavior, using the combined DoDAF and MoDAF profile (UPDM), which consists of a collection of similar views representing several facets of the SoS architecture. As SoS continues to operate, the changes have to be made to the SoS while it is still working, in the (2) operate the SoS activity. In this later, models are used for tracking, evaluating, and predicting operations. Evaluation of the real SoS provides guidance to refine the models. Thus, the model SoS behavior and operate the SoS activities are closely coupled. In the (3) define potential needs, the SoS Manager identifies, at SoS level or CS level, new or changed potential requirements using the UPDM capabilities and operational views that structure the identity and relationship of capabilities and characteristics. In response to new or changed potential needs, the architecture may change in response, in the (4) analyze possible architecture changes activity, the SoS manager refines models and analyzes possible changes in the architecture. After the generation of new architectures, both of of UPDM models and CS models could be associated to create a joint time-based simulation of the whole SoS. The simulation enables the SoS manager to evaluate the SoS arhitecture. Finally, the SoS manager implements SoS revisions by influencing CS owners to update their systems for the benefit of the SoS, in the influence and implement changes activity. Note that the DANSE methodology requires an SoS manager with some authority for the SoS in addition to the SoS architect, the two stakeholders must have knowledge and visibility of the SoS goals. Furthermore, the methodology is based on two key challenges, the dynamicity which represents the constant change and evolution of the SoS, and the emergent behavior, which is the behavior that results from the interaction of CSs.

2.3.5.2 COMPASS approach

The COMPASS (Comprehensive Modeling for Advanced Systems of Systems) project [1] was intended to provide and evaluate model-based methods and tools for development and analysis of SoS models. The COMPASS approach is shown in Figure 2.11, it provides various description levels, beginning with a SysML graphical view which is simple to understand for SoS stakeholders (49), to the textual COMPASS Modelling Language (CML) (91), a formal language that needs trained stakeholders, and that was proposed to carry out many types of analysis for the SoS. CML is based on well-

[1]European funded project. Web site: http://www.compass-research.eu Accessed 05/07/2020

established formalism and includes particular aspects of SoS, as shown in Figure 2.11, such as contracts for constituent systems. Many different types of analyses can be performed using CML, and some will be presentable at the SysML level. CML's semantic is established using the Unifying Theories of Programming (UTP) (92).

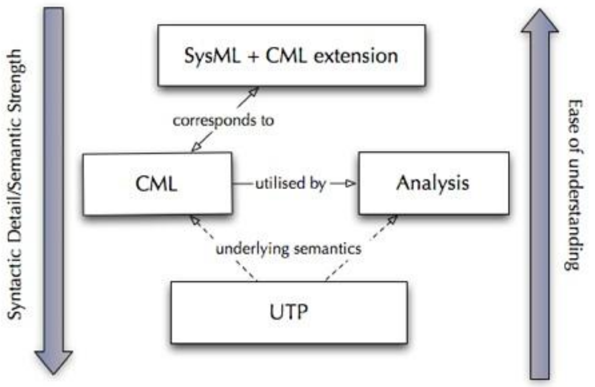


Figure 2.11: The COMPASS Approach (18)

The COMPASS project gave special importance to the development of SoS architecture activity, and to the use of an Architectural Framework (AF) that includes consistency rules defined between the various views produced. That's why the project defined an AF, called the COMPASS Architectural Framework Framework (CAFF) (93). CAFF is based on the SysML language and can be used for defining other architectural frameworks and architectural patterns for SoS. The COMPASS modeling and analysis process for SoS architecture was defined in (19) and applied to an accident response case study in (49). Figure 2.12 depicts the main phases of the COMPASS modeling and analysis process for SoS architecture, which use the SysML and CML languages.

The first step of the process is (1) to determine the SoS requirements and stakeholders, it is followed by (2) the understanding the CSs, their relationships and stakeholders step. As it is difficult to reason directly about the SoS architecture, the COMPASS process proposed to (3) to outline idealistic block diagram, which assumes that all features of all CSs are immediately accessible. Furthermore, it is important to decide how to accommodate each CS, to support the ideal functionalities. The (4) CSs and their components with different groups of functionality, interfaces, and protocols have to be classified, this step helps to (5) improve the idealistic block diagram as well as

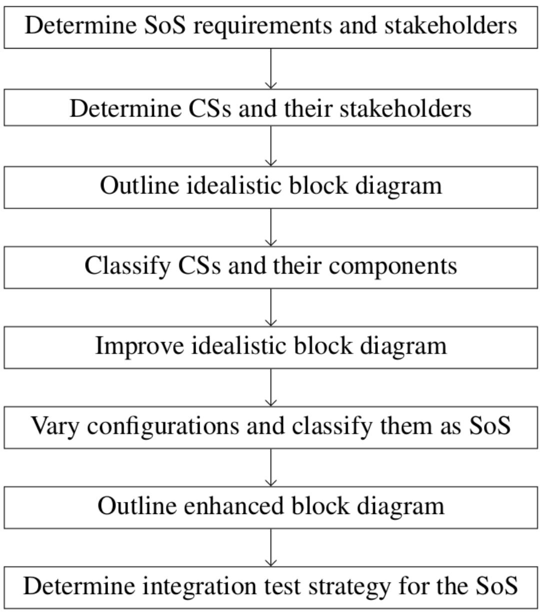

Determine SoS requirements and stakeholders
Determine CSs and their stakeholders
Outline idealistic block diagram
Classify CSs and their components
Improve idealistic block diagram
Vary configurations and classify them as SoS
Outline enhanced block diagram
Determine integration test strategy for the SoS

Figure 2.12: The COMPASS SoS Architectural Modeling and Analysis Process (19)

to (6) vary configurations and classify them as SoS. After the previous analysis steps, (7) a block diagram of the complete SoS is established, The enhanced block diagram must support all configurations. Finally, it is important to (8) determine integration test strategy for the SoS. The project emphasizes that this process is applied iteratively.

2.3.5.3 AMADEOS Framework

The AMADEOS (Architecture for Multi-criticality Agile Dependable Evolutionary Open System of Systems) project [1] goal is to bring knowledge and evolution to the design of SoS, to develop a sound conceptual model, a generic architectural framework, tools, and a design methodology. The previous projects (DANSE and COMPASS) apply SysML approaches to specific viewpoints, deemed essential in providing SoS architecture. The AMADEOS architecture framework benefits from the previous approaches in supporting specific viewpoints, and integrates SysML specific solutions to provide a usable high-level support for designers of SoS. The AMADEOS AF (20) is defined by means of a high-level perspective of activities and artifacts involved in the

[1]European funded project. Web site: http://amadeos-project.eu/ Accessed 03/09/2020

design phases of SoS and by its specialization based on its point of view. The high-level view representation includes four different layers, namely mission, conceptual, logical and implementation.

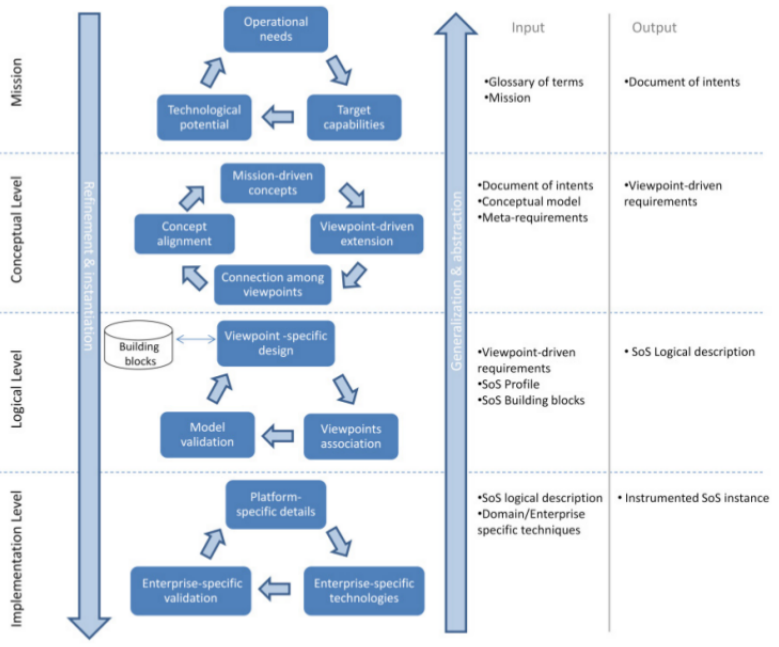


Figure 2.13: The AMADEOS process-based view of the AF (20)

2.3.6 Other SoSE contributions

In SoSE field, there is a multitude of proposals which aims at giving frameworks, tools and methods to model and architect SoS. These different proposals are discussed in the following.

Lock and Sommerville (94) proposed an approach to model SoS throw capability specification of subsystems. The authors proposed a taxonomy for capability concept. Capabilities can be broken down by type (Technical, Socio-technical Resources, Manual, Information Resources, Personnel Resources) and maturity (Current, Legacy, Development). The authors defined a graphical notation to illustrate the dependencies between subsystems throw capability dependencies. The approach is designed to help end-users recognize and evaluate graphically the hazards and associated risks that may occur in complex socio-technical SoS, with specific focus on the role of system dependencies. This work gave some interesting features about capabilities (type and maturity), however, these informations are directly related to the systems that will be used at run-time.

Lu and al. (21) proposed a double layer modeling framework for SoS, based on the complexity analyzing of SoS relationships. The authors hilighted that SoS capability is classified into two types: Capability Requirement (CR) and Capability Property (CP). The CR is the potential ability to execute a specified course of action, while the CP is the inherent ability to execute a specified course of action. The double layer framework addresses a modeling process from micro mechanism modeling to macro behavior distinguishing. As described in Figure 2.14, the key to SoS capability modeling is to analyze micro mechanism on emerging desired capability, and guide the result generated by micro mechanism to predicted capability, which is obtained from complicated relationship, using capability emergence process.

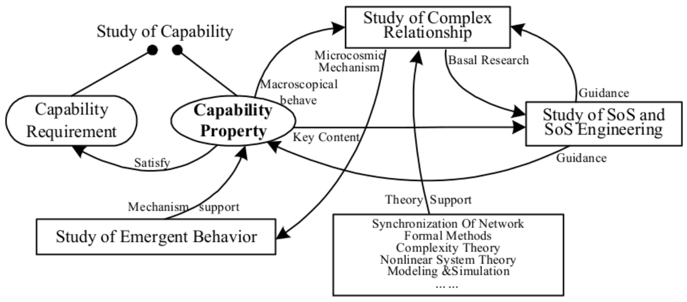


Figure 2.14: Main Concepts of the Double Layer Modeling Framework (21)

(95) proposed a conceptual model for SoS that can support the development of analysis and synthesis tools. The aim of this conceptual model is to reason about some properties of SoS and verify whether specifications are met. In this model, an SoS S = S(R, G, E, CS, B, C, H) is characterized by (1) Roles, R, a set of finite roles the system can assume, (2) Goals, G, representing for each role what the system should do, (3)Environment Model, E, representing the external environment assumptions of the system S under which the system was built to accomplish its objectives in a specific role, (4) A set, possibly empty, of constituent systems, CS, (5) Capabilities, C, representing what the system can do in a given role (6) Behavior, B, the relationship binding the variables (inputs, outputs, states) describing how a system operates, (7) A Strategy, is the decision of a constituent system about its behavior chosen according to its en-

vironment model, and (8) Relations, H, defined on the constituent systems and roles of the System of Systems S. The authors adress a list of relations on the constituent systems, each relation can be represented as a directed graph. The possible relations are: COMMUNICATION (a communicates to c), AUTHORITY (a has authority over c), USE (a uses c), OWNERSHIP (a owns c), COORDINATION (a coordinates with c), OBSERVATION (a observes c), KNOWLEDGE (a knows c). The authors use the conceptual model to reason about degrees of fulfillment of SoS objectives by giving a classification of reasons for failing the SoS objectives, and possible lines of attack to deal with these.

2.3.7 Discussion

When analyzing the state of the art, we found a large number of model-based methodologies, such as OOSEM, Harmony and MagicGrid, which are dedicated to analyze, developing and documenting complex systems. They are widely used to solve different tasks of the systems engineering process, and they have reached a level of maturity, in the identification and gathering of artifacts and best practices for complex systems engineering (14). However, the litterature shows that a SoS is viewed not as a hierarchy built of component systems, but as an environment within which other systems operate, and which can support the addition of new systems that build on systems already in the environment (3, 39). The SoS boudary will be variable, continually adding, modifying, or removing different constituent systems within the scope, and the SE model based methodologies do not follow this reasoning. SE approaches can trace system boundaries and define requirements clearly (81, 96), something that is not obvious in the SoSE. Furthermore, trade analyses and measures of performance allow the optimal allocation of components to requirements while in SoSE, SoS engineers need to take into account considerations beyond the use of existing systems as the constituent systems of these SoS (81). They must allocate the realization details and functionalities that may not be optimal from the SoS point of view. Moreover, SoS engineers do not control the overall development environment of the SoS because the constituent systems must retain their independence.

The AF presented above are just examples among many other frameworks. Indeed, many countries have their own specific framework, most of them are closely related and, therefore, have very similar concepts. However, as argued by (20), AF are “prescriptive” and not “descriptive”, there is still no consensus on offering step-by-step methodological guidance to be followed when using AF to analyze or design SoSs. Therefore, it becomes difficult to make choice about which model or view to use, and it is difficult to take advantage of the interconnected views because none of these views provide a simplified perspective that addresses only the subsets of each view. The architectural frameworks constitute “in depth modeling approach, which requires significant resources” (94).

The COMPASS, DANSE and AMADEOS projects are the approaches presented in the context of research projects. The main advantage of the COMPASS approach, is that it provides a well-defined denotational semantic of SysML blocks by means of CML, that supports a variety of analysis techniques. DANSE proposed methodological guidance and a reduction in the AF according to the target objectives. It focuses on the six models that can be represented as executable forms of SysML, and relies on evolutionary lifecycle based on the idea that SoS grows from interacting constituent systems. The COMPASS and DANSE projects shows the utility of adopting SysML formalisms in order to model different architectural and non-architectural aspects of an SoS. This promotes various forms of analysis and constitutes the first step towards executable artifacts that can be extracted automatically from SysML. In addition, the AMADEOS AF offers an integrated support to all the viewpoints, it offers means to link the high-level perspective to activities and artifacts involved in SoS design phases. The research projetcts relies on iterative lifecycles which focus on identifying new or changed requirements and analyzing possible architecture changes activity. This point helps to improve the SoSE design to manage the SoS problems. However, more recent research argues that in SoSE domain, the design and the end-to-end process and management are required to be balanced (22, 50). In fact, SoS are acquired to satisfy new capabilities in a mission context. The latter is a key element to assist SoS engineers to determine the systems that must be involved and the functions they must perform. It is important to consider the mission thread to bridge the dissociation between the SoS objectives and the individual functionalities undertaken by the systems that constitute the SoS to support the SoS mission. “The allocations of functions or activities to constituent systems can change over the course of a mission thread” (50), and thus, SoS boundaries may change continually dependent on the mission (97).

The SoSE contributions gave interesting ideas, considered as a foundation for thinking differently about SoSE, for example modeling SoS based on capability modeling. These contributions are often based directly on constituent systems capabilities, and does not take into account that systems available at run-time are not really known at early stages of SoS development. However, as argued by Keating (96), “this shift in thinking is an invitation to engage a different paradigm, respective of the unique character of complex SoSs and the different emerging constructs that hold the keys to increasing effectiveness in dealing with SoSs”.

The analysis of the state of the art reveals that a novel approach, with new paradigms, which incorporates the best practices of existing SE approaches, taking into account the SoS artifacts and mission understanding, would be welcomed within the SoSE. From this perspective, we propose in this thesis the maintenance of a mission focus, throughout the SoSE analysis and the architecture process. We propose a mission-based process within the wave model, for acknowledged SoS analysis and design, based on the SysML language. The process take into consideration several features that distinguish a SoS from system.

Chapter 3

MISSION ENGINEERING

In the previous chapter, we presented the state of the art on SoSs modeling and highlighted the need to balance the design and the end-to-end process. We argue that the end-to-end process is embodied in the concept of mission. In fact, Mission Engineering (ME) is the domain that intends to link the engineering activities that are conducted to achieve a mission, with the mission itself. Thus, in order to efficiently use the mission paradigm, we have dedicated this chapter to present the different approaches of ME domain as well as its basic vocabulary. we start by presenting an overview of mission engineering domain. In the later, we introduce its origin, and take up position in the most suitable mission engineering form for our work, because several forms of mission engineering have emerged according to the application domain. We give some preliminary knowledge and the terminology of mission, mission engineering domain, and present the different roles involved in this domain. Then, we describe the different approaches that exist in the literature, which are required to achieve and oversee mission engineering efficiently. The approaches were collected from several sources, mainly the military domain. A discussion is given at the end, to identify key points that have to be considered, to support mission engineering for SoS development process.

3.1 Overview of mission engineering

The following section presents background and definitions of Mission Engineering (ME), in the context of military domain, SE, and SoSE. It defines also the key terms used in this domain, as well as the roles involved in it.

3.1.1 Origin of mission engineering

Mission Engineering (ME) designation has been used for almost two decades mainly in two different domains: Aeronautics and Defense. Even if the principles of ME are the same, its use is definitely different in each domain (22). There are two large and different uses of ME, the first one by the Department of Defense (DoD), and the second one by the National Administration of Aeronautics and Space (NASA). Sousa-Poza (22) highlighted that each type of ME is different because of the essence of the issue being addressed. While there are similarities between the problem and the purpose, the essence of the problem and the way in which it must be handled are somewhat different, if not entirely incompatible. The use of ME at NASA's projects aims to ensure the highest degree of confidence that the expected mission will be accomplished. Thereby, the NASA ME project is highly technical and the solutions must be design-driven, given the nature of space missions. After the analysis and design phases, space missions allow for only limited amounts of adaptation.

In the acquisition and enhancement of DoD capabilities, mission planning plays a key role. Consequently, ME has more recently started to be used in government acquisitions and, especially, in defense projects (98). As supported by (22), platforms in the DoD environment are typically operational. Often, to existing legacy systems[1], engineered artifacts are added. However, legacy systems may evolve continuously, according to the operational environment and the context of use. The consequences are that the exact configuration of components, assemblies, and networks will be uncertain. Thus, ME in this context support approaches that take into consideration the existing operational platform, and give special consideration to the mission to be accomplished purposefully. The emphasis must turn to the purposeful evolution of the network of legacy components towards the intended mission. This reasoning follows exactly the principle of SoSs and remains compatible with our thesis objectives. Moreover, operationally, the DoD acts as an SoS as military commanders bring together forces and systems (e.g., weapons, sensors, platforms) to achieve a military objective (10). That's why the use of ME in our thesis derives from its existing use by the US Defense Department.

More recently, the use of mission engineering becomes more and more widespread

[1]Legacy system is an old or out-dated system, technology or software application that continues to be used by an organization.

in several context. We can find recent papers that apply ME for swarm unmanned systems (99), homeland security as mentioned in (50), and systems engineering (51, 100, 101, 102).

3.1.2 Mission definition

There are numerous definitions for the term "mission". The DoD (103, 104) defined mission as "*the task, together with the purpose, that clearly indicates the action to be taken and the reason therefore*". In (28), mission is defined as "a pre-planned exercise that integrates a series of sequential or concurrent operations or tasks with an expectation of achieving outcome-based success criteria with quantifiable objectives". The authors of (26) defines the mission as follows: "*given a set or type of stimuli, the mission is the collection of tasks, goals and objectives that have to be achieved to successfully address the stimuli. The mission includes all of the physical assets necessary to meet the goals as well as all of the techniques and procedures necessary to effectively employ them*".

The field that emerged owing to the need for understanding, and documenting the end-to-end mission execution, is Mission Engineering (ME) domain. We deal with ME in the following sections.

3.1.3 Mission engineering definition

Mission engineering has several candidate definitions. For example, the US Department of Defense (DoD) defines mission engineering as "*deliberate planning, analyzing, organizing, and integrating of current and emerging operational and system capabilities to achieve desired war fighting mission effects*" (105). Another definition proposed in the DoD Mission Engineering and Integration (MEI) Guidebook (104) defines it as "*planning, analyzing, organizing, and integrating current and emerging operational concepts to evolve the end-to-end operational architecture and capability attributes, including anticipated friendly force and opposing force behaviors, that are needed to inform the communities of interest involved in fulfilling mission needs statements*".

One more definition of mission engineering is the one proposed in (50), it includes a SoSs context, and defines mission engineering as "*the structure of systems engineering and the tactical insights of operational planning to a system of systems to deliver a* *specific capability*". The same source adds that the mission context plays an important role in ME, it is "*a key element to assisting developers and managers to determine which systems have to be involved, what functions they have to perform, and how operators/users will make use of these systems*".

A more general definition is the one proposed by Sousa-Poza (22), where mission engineering is considered as "*a means to bridge the dissociation between mission and the [engineering] activities that are undertaken to support a mission*". The same author claims that ME is an "*approach to coordinate the perspective of the mission owner, the operator, and the engineer*".

3.1.4 Positioning of mission engineering

As argued by Sousa-Poza (22), the key role played by ME, is to integrate the mission that determines the objectives, and the value premise that guides activities. As illustrated in Figure 3.1, there is a dissociation between the purpose and the action, due to the complexity of the problem in general. The operational component includes the engineering activities that will be undertaken. It takes into account legacy systems, artifacts, and assemblies.

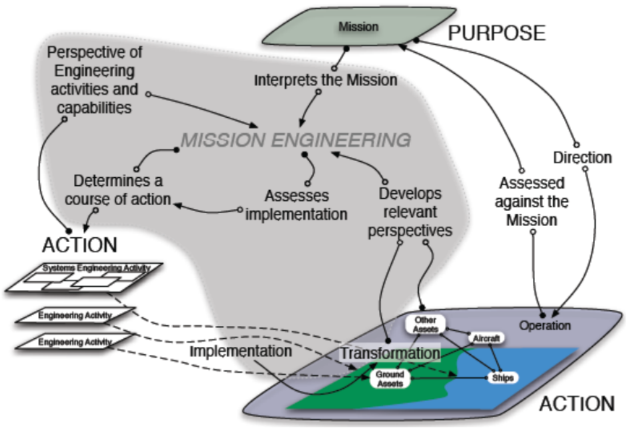


Figure 3.1: Positioning of Mission Engineering (22)

Sousa-Poza (22) affirms that the main role of mission engineering, is to (i) identify relevant perspectives that clarify the mission to be achieved or supported then (ii) form relevant perspectives of each operation. It is important here to consider technical features, legacy systems, mission context, and domain expert input. Finally, (iii) formulate an engineering perspective of activities that are being conducted. Continuous risk assessment, and performance measures must be done too.

The proposition of the use of ME in SoS context is quite recent (106). It began in the military field with the DoD. Legislation for the DoD to establish Mission Integration Management (MIM) activities was included in the 2017 National Defense Authorization Act (NDAA). Six areas of responsibility for establishing MIM activities were identified, including research and development, systems engineering, mission-driven requirements, experimentation, exercises, and Combatant Command coordination (107). ME is the overarching engineering approach for MIM implementation (23). Conceptually, ME seeks to integrate material solutions into a SoS architecture that supports the specific operational mission (See Figure 3.2). It goes beyond data exchange among systems to address cross-cutting functions, end to end control and trades across systems (23).

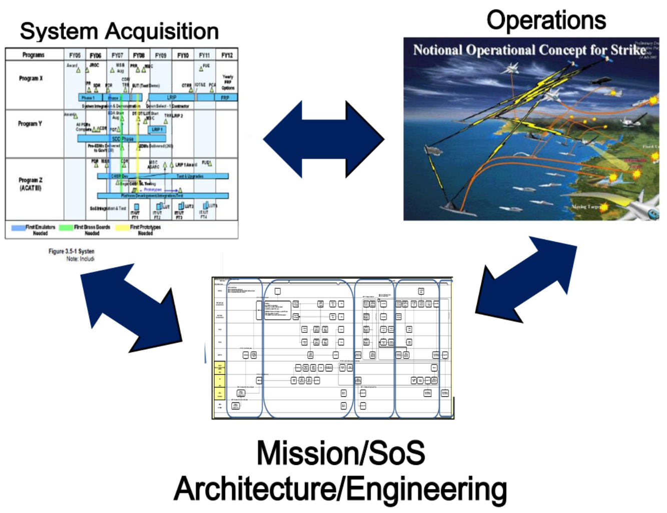


Figure 3.2: Positioning of Mission Engineering (23)

3.1.5 Mission engineering roles

According to Sousa-Poza (22), ME would commonly require the participation of several stakeholders as shown in Figure 3.3. ME implies the incorporation of three key roles' perspectives: **mission owners, domain experts, and engineers** (See Figure 3.3).

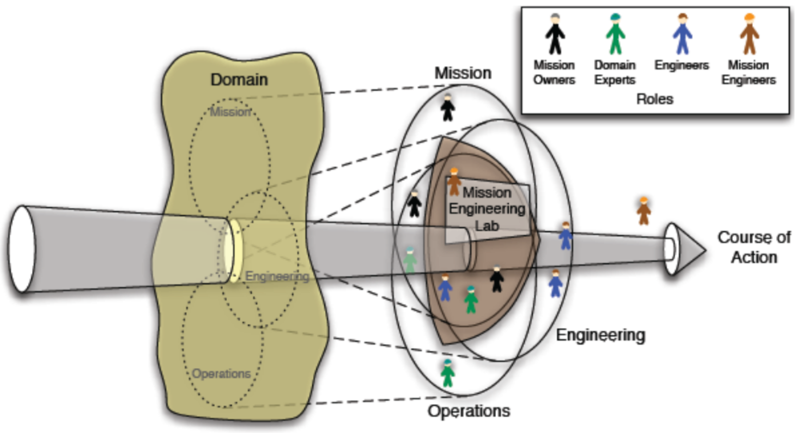


Figure 3.3: The Mission Engineering Involved Roles (22)

Mission owners are responsible for providing the mission decomposition information, including capabilities, standards, and conditions (108). These informations serve to identify the assets being able to meet the needs. The **domain experts** have knowledge of the domain, they provide inputs continuously. Operators, users, and other diverse stakeholders are examples of domain experts (22). The **engineers, systems engineers, and other acquisition specialists** are responsible for designing, developing, and implementing new technologies, artifacts, and assemblies to sustain the operation, by assuring mission capabilities (22). In (108), this role in called "**resource providers**", and are responsible to provide all assets needed to provide mission capabilities. The collaboration space in which ME arises, should bring together the different roles. This may help to provide input, guidance and oversight to the ME decisions (22).

3.1.6 Mission engineering key concepts

Several concepts have emerged in ME field, many of them arise from the military domain. As already mentioned, the military is the first domain who argued for the application of ME in SoS context (106). In the cited domain, ME is considered as the SOI, and applies systems engineering processes and knowledge to the design of missions (109).

"**Mission environment**" is defined by (26) as "*all of the entities and interactions, conditions, circumstances, and influences involved in the prosecution of the SoS against the mission*". The **mission environment** is populated with **mission threads**. These **mission threads** are "*the description of the end-to-end set of activities and component systems employed to accomplish specific subsets of the mission goals and objectives*". Finally, the **context of the mission** includes the **operational environment (Mission Environment)** that contains the natural environment, external stimuli, the factors that may influence the mission and other interactive entities (26, 110). In SoS domain, the mission context is unlike that in traditional SE approaches, in which there is little flexibility because individual functions are mapped to only one element in the system (50). According to (50), "mission context is a key element to assisting SoS engineers to determine the systems that must be involved and the functions that they must perform".

3.2 Mission engineering approaches

The following section presents related research that proposed mission engineering approaches, and discuss their applicability in SoS context.

3.2.1 Military mission and means framework

The Missions and Means Framework (MMF) (24) is a framework for explicitly specifying the mission, allocate means, and assess mission accomplishment (See Figure 3.4). The MMF consists of 11 elements used to define military operations. As shown in Figure 3.4, seven levels specify the mission: (1) interactions and effects that describe "how" the course of actions changes the state of components, (2) components and forces that defines the "by whom" specification, represented with the military actors (integrated units, personnel, equipment,.etc), (3) functions and capabilities define the capabilities which enable forces to conduct operations, (4) tasks and operations that define the "do what" of the mission, and describes the implied tasks for mission accomplishment. The purpose of this level is to analyze the task outputs and subsequently evaluate the mission effectiveness, (5) index and location/time that define the where in terms of geographic location and the when in terms of time, (6) context and environment that define under what circumstances a mission is to be accomplished, and (7) mission purpose that defines the why of the military evolution and indicates the reason and purpose of the mission.

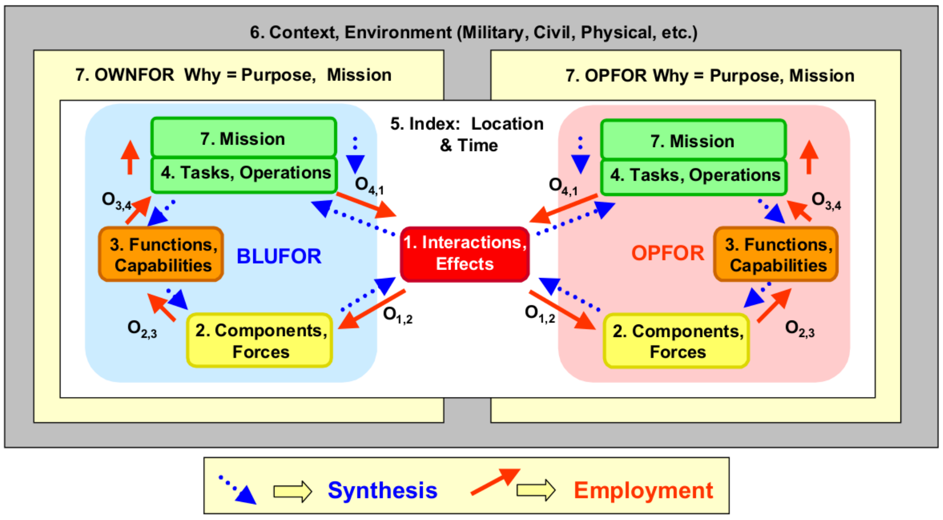

6. Context, Environment (Military, Civil, Physical, etc.)
7. OWNFOR Why = Purpose, Mission
7. OPFOR Why = Purpose, Mission
5. Index: Location & Time
7. Mission
4. Tasks, Operations
3. Functions, Capabilities
2. Components, Forces
1. Interactions, Effects
BLUFOR
OPFOR
$O_{1,2}$
$O_{2,3}$
$O_{3,4}$
$O_{4,1}$
Synthesis
Employment

Figure 3.4: The Missions & Means Framework (24)

In addition to the seven levels, the following four operations are included in the framework, each operation works in two directions (Synthesis or Employment): (i) $O_{1,2}$: transforms interaction specifications (Level1) into component states (Level2) and conversely, (ii) $O_{2,3}$: transforms component states into functional performance (Level3) and conversely, (iii) $O_{3,4}$: transforms functional performance into task effectiveness (Level4) and conversely, (iv) $O_{4,1}$: transforms task sequences into interaction conditions and conversely.

Authors of (25) considered that MMF is a generic approach for modeling mission planning and execution, in which specialized domain knowledge becomes necessary for expressing individual levels and operators. Therefore, the authors proposed an ontology to express the MMF framework, intended to express MMF's complete Level and Operator set from both mission Synthesis and Employment perspectives (See Figure 3.5).

3.2.2 Approach to Mission-Level Engineering of SoS

The authors (26) proposed a class of Systems of Systems (SoSs) called Mission-Level Systems of Systems (MLSoS). They specifically explore and propose a solution to the

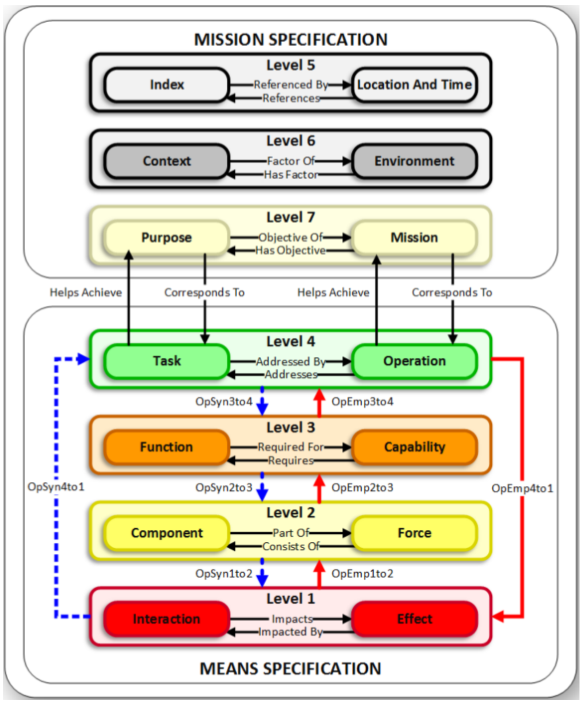


Figure 3.5: Conceptual Diagram for the MMF Ontology (25)

integration and interoperability (I&I) problems, facing mission-level SoSs engineering in the military domain. This work, belonging to the military domain, is one of the few works that considers the mission level in SoS development. In (26), the authors presented a modeling approach to define an SoS/mission architecture using both a physical space and multiple event space constructs. This approach is based on the fact that SoSE, at mission level, must use inputs from the user, at every step of the process. Thereby, the MLSoSE process must be cyclic. Figure 3.6, shows the cyclical nature of the MLSoSE, and reveals that the emphasis here is on integration, interoperability and employment of the SoS.

The main step of the process is to define mission environment and mission thread (respectively ME and MT in Figure 3.6). To delimit mission environment, the authors proposed to restrict the architecture to only items that can be found within the operational context, and thus the mission environment is whittled down. From this, a collection of events is formed, within the physical space, which constitutes a mission thread or event space. To define the SoS architecture, the authors proposed a

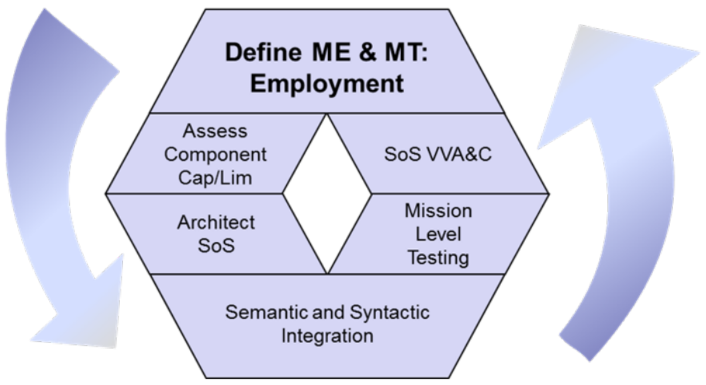


Figure 3.6: Mission-Level SoS Engineering Process (26)

top-down approach. The SoS is at the top, and the challenge is to describe events and physical entities at the needed abstraction level. To provide a consistent level of abstraction of the conceptual model, the authors suggested the use of three abstraction paradigms, that define an adequate abstraction for the SoS. These are entity abstraction, functional abstraction, and I&I abstraction. Physical entities are grouped into five major functional groups: sensors, actors, controllers, stimuli, and environment. The authors used the of an Observe, Orient, Decide, Act (OODA) loop first proposed by Boyd (111, 112), to address functional or event abstraction. Finally, the concepts of Autonomy, Collaboration and Cooperation (ACC) (113) on OODA are used as an I&I abstraction.

The main idea addressed by the authors to model the mission, is to explicitly investigate the interactions among the components, and determine how those interactions may drive the behaviors of the SoS. To do this, they proposed the use of Directed Acyclical Graph (DAG), to capture the edge events from the physical space multi-graph, into a pseudo-linear DAG (114). Thus, the interstitials and the causality of physical space may be captured explicitly, in a form that is readily operable.

In order to measure the efficiency and behavior of the SoS appropriately, the authors introduced two mission-level metrics that describe the performance, behavior, integration, and interoperability. The first one is the probability of success, which is the likelihood of achieving the objectives of the SoS. The second one is the breaking point, which is the number of stimuli, that act on the SoS, and which cause the SoS to stop working, or the probability of success to fall below an appropriate threshold. To address explicitly and quantitatively syntactic and semantic I&I (See Figure 3.6), the authors proposed the use of a set of complementary metrics, built on the previous metrics: the probability of realization, level of integration, and degree of interoperability.

3.2.3 Mission engineering within the systems engineering ’V’ Model/ Mission Engineering Integration and Interoperability (I&I)

In the latest five years, the US Navy was focusing especially on integration and interoperability (I&I) in ME. I&I is a Navy-wide project to understand how well current systems fulfill the Navy mission, and to find gaps. In order to enhance mission efficiency, the Navy seeks to understand integration / interoperability issues between systems and identify where investments are needed. In this context, Moreland (27) proposed to incorporate a mission focus into the traditional SE “V” as depicted in Figure 3.7.

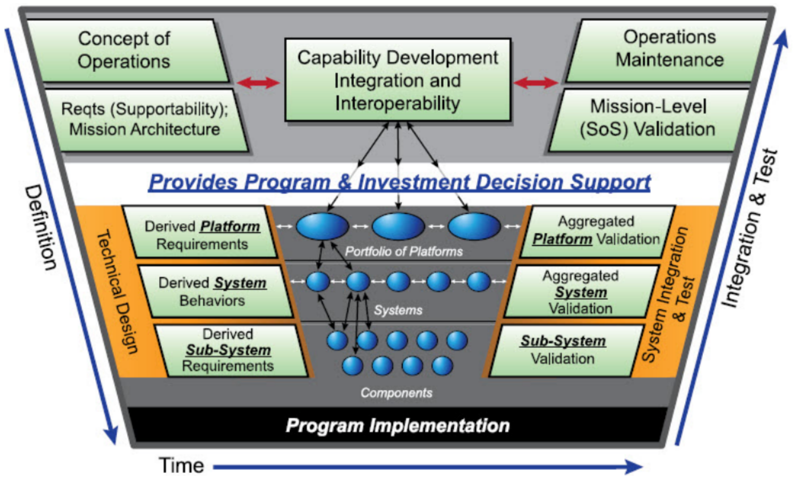

Concept of Operations
Reqts (Supportability); Mission Architecture
Capability Development Integration and Interoperability
Operations Maintenance
Mission-Level (SoS) Validation
Provides Program & Investment Decision Support
Definition
Technical Design
Derived Platform Requirements
Derived System Behaviors
Derived Sub-System Requirements
Portfolio of Platforms
Systems
Components
Aggregated Platform Validation
Aggregated System Validation
Sub-System Validation
System Integration & Test
Integration & Test
Program Implementation
Time

Figure 3.7: Mission Engineering within the SE “V” Model (27)

The emphasis in this approach is on capability development, because the individual systems that comprise the capabilities are inherently flexible, functionally overlapping, multi-mission platforms supported by a complex backbone of information communication networks (110, 115). As shown in Figure 3.7, the mission layer is added to the classical SE process. In the ME layer, the capability development I&I is considered as a cross-cutting engineering activity that provides the basis for defining systems, evolving systems and performing integration and test. It helps to bridge mission effects and SE. For the ME layer, the operational concepts drive mission capabilities. These latter, in turn, drives the mission requirements, these translate to system requirements and component functions and end items.

3.2.4 Enterprise Strategic Planning for Engineered Systems

In his book, Wasson (116) presents the organizational mission engineering process for systems in context of SE. The process is to be used in the context of higher-order systems, such as corporate enterprise management, shareholders, and the general public, have an expectation that short- and long-term gains (survival, benefits, return on investment, etc.) are obtained from developing an enterprise that meets the needs of the consumer. The author highlighted that an organization acquires systems to support strategic and tactical objectives. The operational need is fundamentally embedded in recognizing the vision and mission of the organization. Figure 3.8 illustrates the process, it consists of two main loops the strategic planning loop and the tactical planning loop (See respectively (1) and (7) in Figure 3.8).

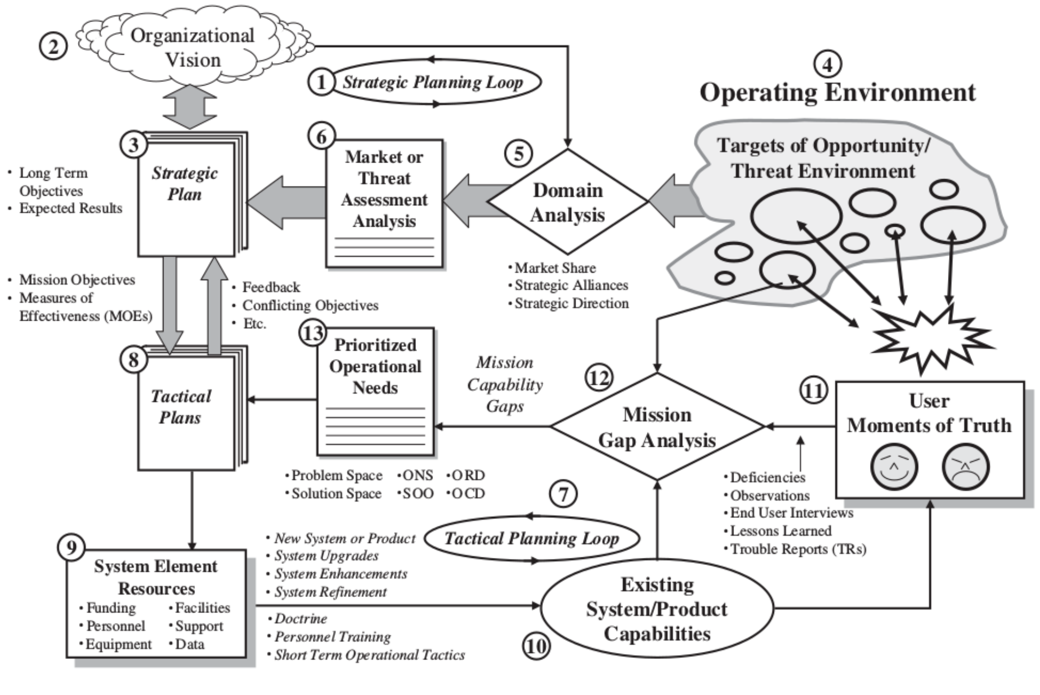


Figure 3.8: The Organizational Mission Engineering Process within the Enterprise (28)

The strategic planning loop: with an organizational vision (See (2) in Figure 3.8), the foundation for long-term organizational development starts. The ME process starts with the analysis of the OPERATING ENVIRONMENT composed of targets of opportunity and threat environment (See (4) in Figure 3.8). The resulting report is associated with long-term organizational vision of what is to be done, which constitute the basis for developing the strategic plan. The strategic plan (See (3) in Figure 3.8) outlines the expectations for the next 5 years. It describes a set of long-term objectives, each of which should be realistic, measurable, and achievable.

The tactical planning loop: after establishing the strategic plan, the mission analysis step began. The aim is to develop a tactical plan (See (8) in Figure 3.8) that will describe HOW to satisfy the mission's long-term objectives. Incremental and short-range tactical strategies, that elaborate near term mission objectives, have to be developed and sustained. Measures Of Performance (MOPs) quantify the performance needed for each tactical objective. In order to achieve the goals, the MOPs serve as metrics for comparing expected and real mission accomplishment progress. Each organizational element proposes a tactical plan, in response to the tactical objectives, which explains how the organization's leadership envisages fulfilling the MOPs's objectives. The tactical plan specifies the ways the Organizational System Elements (OSEs) such as PERSONNEL, FACILITIES, EQUIPMENT, PROCEDURAL DATA, and MISSION RESOURCES will be deployed, managed, and assisted (See (9) in Figure 3.8). It may necessitate the development of a new system, product, or service, or the upgrade and enhancement of existing ones. An objective, and introspective evaluation of the current system/product capabilities, is part of successful mission gap analysis. It is important to carry out objective, unbiased and realistic assessments of system/product capabilities (See (10) in Figure 3.8). An important phase of the process is to perform mission gap analysis (See (12) in Figure 3.8). Mission gap analysis aims, on one hand, identifying the gap between the actual product/service capability of the organization, and the objectives to be achieved. On the other hand, it aims at collecting deeper insights about strengths, weaknesses, threats, and opportunities of the actual plan. The results of the mission gap analysis are documented as prioritized operational needs (See (13) in Figure 3.8). User moments of truth (See (11) in Figure 3.8) allow collecting real-world field data, that constitutes a degree of risk that has an effect on the organization, resources, or human life. Wasson (28) precises that it is important to collect moments of truth data, through interviews and user feedback, or user community surveys. This will allow to gather best practices, user experiences, and lessons learned.

3.2.5 M2Arch: Mission-Based Methodology for Designing software-intensive SoS Architectures

There is contribution in software intensive SoS where Silva (30) proposed M2Arch, a model-based refinement process for SoS architectural modeling that uses missions as the starting point. Mission model is defined using mKAOS (29), an SoS mission description language. In mKAOS, mission is a specialization of goal to SoS domain. The mission is refined with and/or operators until finding sub-missions that can be handled by a constituent system. The authors defined SoS mission as encompassing five concepts: (i) priority, (ii) trigger, (iii) constraints, (iv) parameters, and (v) tasks, that are functional operations to be executed (See Figure 3.9).

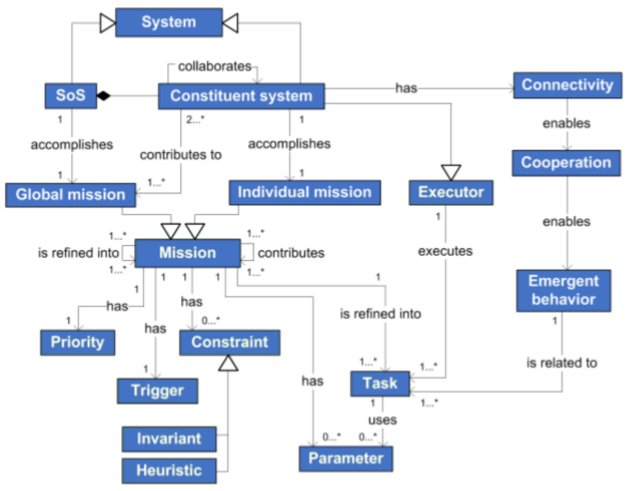


Figure 3.9: Conceptual model for missions in System of Systems (29)

As depicted in Figure 3.10 three steps characterize the M2Arch process. In the (1) definition step, a mission model is defined using mKAOS, and the structure of the architecture is generated. Later, when the architect specifies the behavioral aspects of the SoS, the concrete SoS architecture is generated. It will be (2) validated and (3) verified in the two steps that follow the mission modeling step. The validation is done using a simulation based approach while the verification uses statistical model checking to verify whether specified properties are satisfied.

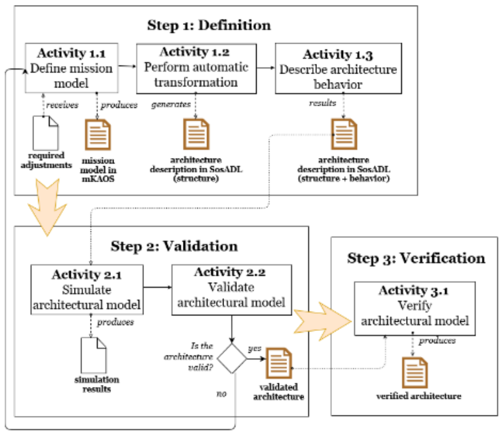

Step 1: Definition
Activity 1.1 Define mission model
Activity 1.2 Perform automatic transformation
Activity 1.3 Describe architecture behavior
receives
produces
generates
results
required adjustments
mission model in mKAOS
architecture description in SosADL (structure)
architecture description in SosADL (structure + behavior)
Step 2: Validation
Activity 2.1 Simulate architectural model
Activity 2.2 Validate architectural model
produces
simulation results
Is the architecture valid?
yes
no
validated architecture
Step 3: Verification
Activity 3.1 Verify architectural model
produces
verified architecture

Figure 3.10: M2Arch Process (30)

3.3 Discussion

Mission engineering is a recently proposed concept that needs methods, tools, and practicable means to implement (51). Furthermore, it requires a detailed understanding of the operational view of the system. Recently, the ISO/IEC/IEEE 21839 (52) has stressed the importance of considering the mission, and its context, in the development life-cycle of systems.

Several approaches for ME have been presented in the chapter. They trace the main steps needed to achieve mission engineering successfully. In the approaches that aim at applying ME in SE context, ME layer is added to the SE life-cycle, a mission has a number of objectives, which are translated into system requirements and component functions. These approaches facilitate the creation of efficient ME solutions, using familiar languages, and processes of SE community. However, these practices do not fully address the unique characteristics of mission engineering, which is to address the end-to-end mission as SOI. A mission has a goal, which is achieved through a sequence of operational activities, ME determines those operational activities and allocates them to operational nodes for execution. The operational activities are allocated to different systems, and thus to SoS. It is therefore interesting to consider the mission in an SoS development life-cycle. That's the objective of our thesis.

On the other hand, the approach of (26) for MLSoS put a lot of emphasis on I&I. As already mentioned in section 1.2.5, engineering in the military domain focuses mainly on I&I. This has the advantage of the construction of mission-oriented systems of independent and highly interoperable systems. Furthermore, (26) included mission-level metrics to continuously measure and assess mission performance. However, in this work, the architecture of the SoS is not guided by operational activities as recommended in ME. Indeed, this work is based on the investigation of the interactions among the systems, that form the mission environment, to determine how those interactions may drive the behaviors of the SoS. The process targets the interactions, rather than the mission or operations. In our thesis, we want the mission to be the guide and the controller of the choices during the design and evolution stages of SoS.

In order to take full advantage of ME, we propose in this thesis to incorporate ME process in the SoS life-cycle. The first chapter has shown that model-based approaches plays prominent role in the domain of SoSE. Thus, we propose in the next a SoSE approach that integrates recent trends in MBSE, and that is based on the ME principles.

Chapter 4

MISSION-ORIENTED PROCESS FOR SYSTEM OF SYSTEMS ENGINEERING

In the chapter 3, we gave an overview on the state of the art related to SoSE methods, that can be used to create and manage SoSs. We stressed the role played by the MBSE in the design and implementation of effective SoSs. We highlighted that new research (22, 50) recommends balancing the design of SoSs and the end-to-end process, and to bridge the dissociation between the SoS objectives and the individual functionalities undertaken by the CSs. In order to meet this recommendation, we described in the chapter 3 different approaches that we found in the ME literature. ME intends to link the engineering activities that are conducted to achieve a mission, with individual functionalities undertaken by different nodes. The ME state of the art has shown us that the existing approaches do not fully address the unique characteristics of ME, which is to address the end-to-end mission as SOI. Moreover, the most existing approaches were proposed in the context of SE, rather than in the context of SoSE. Knowing that the operational activities are allocated to different systems in ME, and thus to SoSs, we believe that SoSE is the most appropriate context to ME.

In order to link SoS objectives to the individual functionalities undertaken by the CSs in SoSs context, we consider that SoSs are acquired to satisfy new capabilities in a mission context. The latter is a key element to assist SoSs engineers to determine the systems that must be involved and the functions they must perform. Indeed, in this perspective, this chapter aims at defining a process to build and evolve SoSs, that is based on the integration of ME concepts and MBSE fundamentals, called MOP-SoSE (Mission Oriented Process for System of Systems Engineering). Such a process must be described by concrete steps with corresponding guides to specify the SoS mission and the base models. Furthermore, it is important to identify the role of participants in such a process. To do that, we firstly present a global overview of the process. Then, we enumerate the different roles involved in the process. Next, we propose a conceptual model illustrating the different concepts used in the process. Afterward, we go into detail about each step and show the associated models in the SysML language, using an illustrative example. Finally, we conclude the chapter with a discussion.

4.1 Global overview of the MOP-SoSE

This section shows the general approach proposed to incorporate mission engineering, within the SoSE life-cycle. It was already detailed in Cherfa et al. (117). In the beginning, we introduce the key points of our process to position it. Then, we present the MOP-SoSE life-cycle, which is adapted from the SoSE wave model (11). Afterward, we illustrate the general process and its main steps. Finally, we introduce the illustrative example, which allows us to simply explain our contribution along the chapter.

4.1.1 MOP-SoSE key points

The key ideas on which the process relies are as follows: (i) The process is applicable to acknowledged SoS, in which the organization manages the SoS, and supports the SoSE. Independent organizations and SE teams are responsible for the constituent systems. (ii) An SoS is considered as an environment within which CSs operate, to accomplish a given mission, rather than a hierarchy built of component systems. Thus, the SoS environment is uncertain, because over the course of a mission thread, and according to mission context, the allocation of functions to CSs can change, and then different CSs may be added, removed, or modified in/from the SoSs. (iii) Mission context determines mission thread, and then mission context helps to determine the functionalities needed, and then the CSs to be involved. (iv) The end-to-end mission is considered in our process as SOI, from which architecture is generated as automatically as possible, to avoid information loss between the application domain expert and the system architect. (v) The concrete architecture is elaborated from the abstract one. The latter serves as an invariant that guides the choices of concrete entities.

4.1.2 MOP-SoSE life-cycle

For the development of the MOP-SoSE life-cycle, we relied on the wave model proposed by Dahmann et al. (11), and we adapted it for ME principles and terminology. We chose this model because its driving features that were cited in (11), embody several ME's attributes. These attributes are as follows: **(i) multiple overlapping iterations of evolution**: illustrate the fact that to accomplish a mission, an SoS needs to leverage developments of its CSs. **(ii) ongoing analysis**: mission in context of SoSs requires continual analysis, to address the complex nature of the SoS, and mission context uncertainty. **(iii) continuous input from external environment**: is necessary to engineer mission, as every SoSs operator has control just on a limited part of mission environment. **(iv) architecture evolution**: over the course of a mission thread, and following a mission environment change, the SoS's architecture may evolve. Thus, it is important that the expected mission-oriented architecture be incrementally implemented.

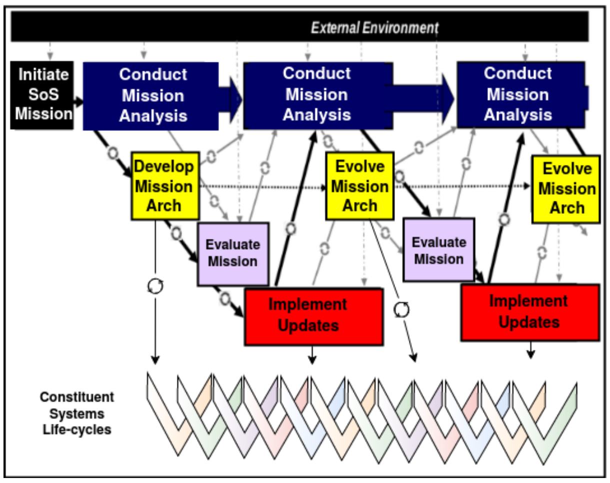


Figure 4.1: MOP-SoSE Life-cycle adapted from (11)

To provide all the assets required for mission capabilities, negotiations with CSs are required. These negotiations can results by the implementation of new artifacts at CSs level. To illustrate this aspect, we represented the CSs life-cycles on the wave model. Engineering of CSs is performed in conjunction with SoS engineering, and the evolution of the SoS depends on updates in the CSs within their own life cycles (See Figure 4.1). The five MOP-SoSE steps are depicted in Figure 4.1: (i) initiate mission, (ii) conduct mission analysis, (iii) develop mission architecture, (iv) evaluate mission, (v) implement updates. The engineering activities contained in step are described in the next section.

4.1.3 MOP-SoSE roles

In the SoSE development environment, “two levels of stakeholders exist with mixed, possibly competing interests: the SoS stakeholders and constituent system stakeholders” (10). Since the constituent systems are independent and have their own objectives, stakeholders of individual systems may have little interest in the SoS, may assign SoS needs low priority, or may resist SoS demands pertaining to their system (10). To manage the competing stakeholder interests, it is important for SoSE engineers to focus on the mission and the operational view of the SoS, and to balance the SoS objectives with the CSs objectives (10). In the following, we describe the two levels of stakeholders.

4.1.3.1 SoS level stakeholders

We argue for the definition of three stakeholders to be involved in the MOP-SoSE at SoS level: mission owner, application domain expert (ADE) and system architect. The state of the art on ME has shown that the participation of the mission owner and the ADE is required. As our approach considers ME in the context of SoSs, we support the use of another stakeholder, which is the system architect, so that he integrates heterogeneous CSs, and realizes the architecture. In what follows, we explain the participation of each role.

- **Mission owner:** the mission owner has precise knowledge of his own mission. Thus, she/he is responsible for providing information on mission decomposition, including priorities, risks and critical paths. Furthermore, she/he is responsible for defining the different metrics that allows to evaluate the mission effectiveness.

- **Application domain expert:** the application domain expert (ADE) masters the domain knowledge. Therefore, through her/his experience, this expert can anticipate the solution when refining the mission. She/He is responsible for defining the mission thread, and focuses on mission context. The ADE has the necessary knowledge to balance the SoS mission with constituent system goals, even if she/he does not control the CSs that impact the SoS. Moreover, the ADE have to provide inputs from mission environment continuously.

- **System architect:** the system architect is responsible for the generation and realization of the architecture. Based on the different models realized by the ADE, she/he is responsible for deploying the required constituent systems to produce a concrete architecture. If needed, she/he must define the SoS level new requirements and artifacts for CSs, which are necessary for the integration of CSs.

4.1.3.2 CSs stakeholders

The MOP-SoSE's CSs level is characterized by the participation of different types of **engineers and systems engineers**, they represent the various stakeholders involved at the level of CSs life-cycle, and working to meet CSs local goals. But in the same time, they are responsible for designing, developing, and implementing new artifacts, to support the SoS mission. The new artifacts are negotiated with the SoS architect to find a good compromise between CSs goals and the SoS goal, since they can be conflicting. The point to emphasize is that since our process is applicable for acknowledged SoS, there is active cooperation between SoS and CSs. Thus, CSs engineers must be aware of the SoS requirements, in order to be able to predict and reason about the impact within the encompassing system (the SoS), when they carry out change at CSs level, as argued by (118).

4.1.4 MOP-SoSE general process

Figure 4.2 illustrates our process. It consists of different engineering activities, and involve several stakeholders. The process offers a disciplined procedure for explicitly specifying the SoS end-to-end mission and generating the appropriate architecture. It is composed of top-down planning and decision making, and bottom-up adjustments.

The process aims to refine the mission, until the architecture is reached, while preserving the mission traceability. Therefore, the refinement activities are as follows:

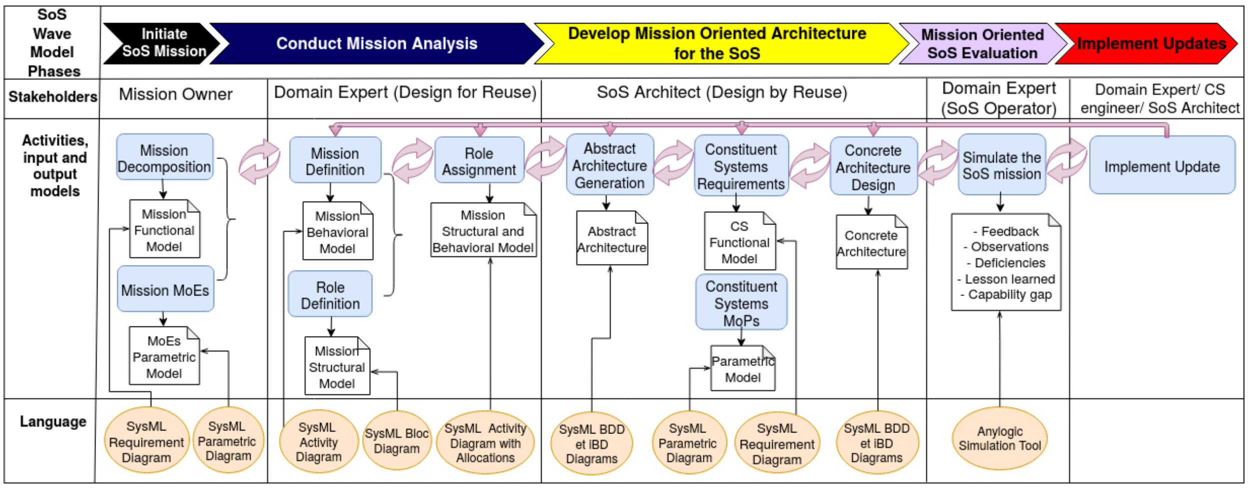


Figure 4.2: Actors and Responsibilities of MOP-SoSE

1. Mission decomposition: this activity is intended to provide a functional coarse grain view of the mission. This aspect is achieved through an analysis of the general mission objectives to recursively identify more precise sub-mission objectives. The criterion for stopping the mission decomposition is the identification of a process that can perform a given sub-mission. We have developed a profile extending the SysML requirement diagram to refine the main mission into sub-missions, create context-dependent variation points, and capture mission risks. Therefore, this step results in a mission functional model of the SoS.

2. Mission Measurements of Effectiveness (MoEs) definition: in this activity, the mission owner defines effectiveness measures, using the SysML parametric diagram. These measures will be used as metrics to assess the overall performance of the SoS mission.

3. Mission definition: the aim of this activity is the design of the operational view of the mission. It consists on the definition of mission threads and activities. It results on a fine-grained behavioral view of sub-missions using *activities*.The view is elaborated using the SysML activity diagram. Each sub-mission in the mission functional model is associated with an activity using the refine relationship. Complex activities can be decomposed into sub-activities, and the criterion for stopping activity decomposition is when a sub-activity corresponds to a role capability that we call *action*.

4. Role definition: the *role* is used to provide an abstract representation of hierarchy of entities having capabilities that enable the achievement of the mission. The capabilities could be provided or required by roles, thereby allowing the composition of roles. The produced model for role definition is based on a SysML profile extending the block definition diagram.

5. Role assignment: as mentioned previously, activities are composed of actions that correspond to role capabilities. The role is composed of several capabilities, and the same capability can appear in different roles. Therefore, this step is intended to designate the role that must be associated with each action of an activity. This association creates a link with the constituent system through the assigned role.

6. Abstract architecture generation: the architecture is a structural view that describes the constituent systems of the SoS and their connections. However, all the above-mentioned definitions refer only to roles instead of constituent systems. Therefore, the first generated architecture from the given definitions corresponds to the abstract architecture of the SoS. It is described using both of the SysML internal block diagram and SysML block definition diagram.

7. Concrete systems requirements: before replacing roles with concrete CSs. The architect can identify new requirements at the CSs level, necessary for their integration into the SoS. These new requirements may require negotiation with CSs engineers, and are described using the SysML requirement diagram.

8. Concrete systems Measurements of Performance(MoPs): MoPs are described for each service in the architecture, to determine the capabilities and limitations of all relevant CSs. This helps to choose the best CS to handle a given action. The SysML parametric diagram is used to capture MoPs.

9. Concrete architecture design: the abstract architecture is progressively refined during the architecture analysis to get the concrete architecture. For this activity, both the SysML internal block diagram and SysML block definition diagram are employed.

10. Simulate the SoS: the simulation of models is necessary to evaluate the ability of an architecture configuration, to accomplish the specified SoS mission. It allows

also to confirm performance, and to discover errors.

11. Implement update: updates can be done at the level of the SoS models, or at the level of the CSs. Individual updates within a CS, follow system engineering life-cycle. The ADE has only to influence those changes with the CS engineers.

4.1.5 MOP-SoSE concepts

To avoid any ambiguity, we introduce in the following section a mission conceptual model serving as the basis for our approach. The conceptual model was proposed based on the modeling experience using the SysML language of SE approaches presented in the state of the art section 2.3, on SoS artifacts defined in (10, 34), and on the overall experience in the rigorous definition of a mission, that we found in the works presented in the section 3.2. in the following, we define the involved concepts, and highlight them in Figure 4.3.

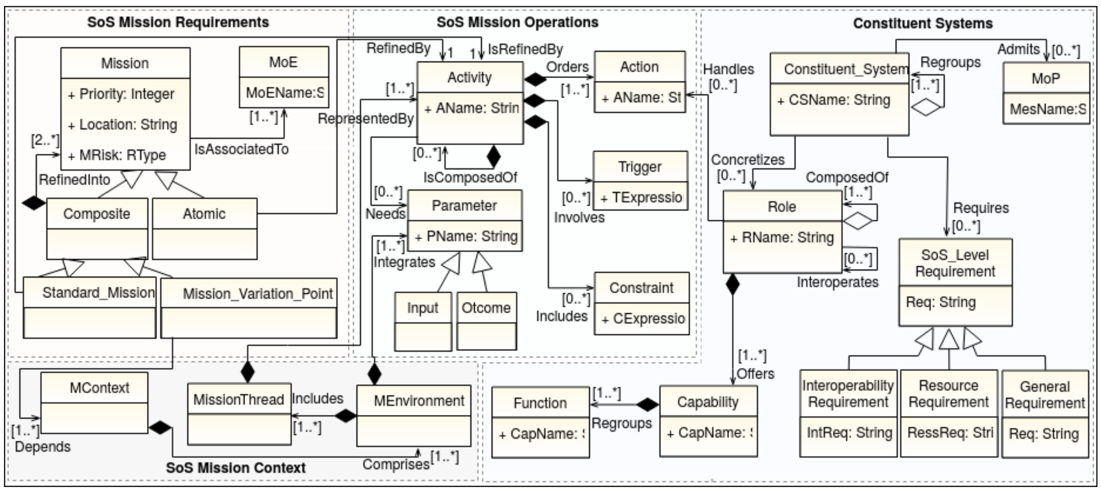


Figure 4.3: Mission Conceptual Model

In our process, we define a **SoS** as "a set of interacting systems that interact with each other and their environment to provide a common mission" (69). The **mission** is the main concept on the conceptual model. We define a **mission** as a finality that the SoS must achieve by collaborating constituent systems. We suggest decomposing the high level mission (generally abstract) into more concrete missions. In the model, this aspect is expressed by the existence of the two classes **atomic** and **composite** and the relationship **refinedInto** between **composite** and **mission**. The refinement is stopped when we can identify the **activity** that is associated with the **mission**. The attributes of the mission class allow the specification of the characteristics of the mission as a location if it is important, risk, etc.

For each mission, **effectiveness measures** must be determined. The relationship **IsAssociatedTo** associate for each mission, the corresponding measures. The **MoE** is a traditional term widely used in systems engineering to describe how well a system carries out a task within a specific context. Likewise, we used the MoE in our process to measure the fitness of a SoS, to fulfill the mission. The MoEs represent non-functional mission objectives for the SoS, expressed in numerical format. It serves as the high level key performance indicators that would be checked when the solution layer is specified.

The nature of the collaboration between composite missions is basically described by the two variants of the mission composite: **standard mission** and **mission with variation point**. **A standard mission** is composed of sub-missions related with the AND, while a **mission with a variation point** is composed of mission variants (OR decomposition), which gives the possibility to several alternatives. The choice of a mission variant depends on the **mission context**. The **mission context** defines the circumstances under which a mission is to be accomplished (24). It encompasses the **operational environment** (MEnvironment in the model), which is defined as "conditions, circumstances, and influences involved in the prosecution of the SoS against the mission" (26). We proposed to represent the **operational environment** as a set of contextual **parameters** that will determine the course of actions to be performed. The **operational environment** may evolve over time, thus, the parameters are always updated.

As illustrated by the relationship **includes** in Figure 4.3, the **mission environment** is populated with **mission threads** (26). These latter are the description of the end-to-end set of activities that serve to accomplish specific subsets of the mission goals and objectives. An **activity** orders a set of **actions**; it can regroup **triggers** and **constraints**; and it can require **input parameters** and provide **output parameters**. The **Role** handles **action** and gathers the required competences (**capability** concept) to play the role needed to accomplish the action. The **capability** of a role is defined as "the ability to provide some expertise to the wider needs of an SoS" (94). It is formed through the integration of several **functions**. We define a **role** as "an abstraction of the characterization of the ideal behavior that will fulfill an action" (119). It inter-operates with other roles (communicate, exchange data, etc). Interoperability mechanisms can be present in an SoS; therefore, interoperability is possible by sub-typing the communication media or by indirection via another CS.

Several types of **constituent systems** could be used to concretize a role in the concrete architecture: humans, processes, hardware and software systems, and institutions. A **constituent system** is chosen when its capabilities match those required by a role, and by considering the **performance measures**. MoP "measure the performance, or more generally the properties, of a system, component or service and can be compared with its own specification and with other systems that perform the same function(s)" (31). The constituent systems is integrated to meet a role capability. Measures must be defined for each CS to guide the choices.

Due to the independence of the CSs, it is possible that when integrating a CS at the SoS level, new **requirements** are needed. These latter can represent a need for **means of interoperability**, for a **resource**, or other **general requirements** (as a need of new functionalities).

4.1.6 SoS example used for illustration

To illustrate the different steps, and activities of the MOP-SoSE, we took a classic example of the SoS, widely used in the literature, which is Emergency Response System of systems (ERSoS). The latter is a widely used example of SoS (18, 36, 49). An ERSoS is responsible for providing emergency aid on demand to members of the public. This SoS encompasses existing agencies (such as fire, police, hospital) with independently owned and managed systems nevertheless collaborate to deliver a service on which reliance is placed (36). ERSoS principal mission is to give the emergency response in case of major incident.

4.2 MOP-SoSE engineering activities

In this section, the whole MOP-SoSE process's activities, described in Figure 4.2, are detailed. For each activity, we describe the related steps, and the way the models could be specified in SysML. We adopt intentionally simplified examples on ERSoS, to simply illustrate the process. As stated earlier, the activities are associated with the wave life-cycle presented in Figure 4.1.

4.2.1 Mission Decomposition

The first activity of the process, we suggest, should be devoted to the functional modeling the mission. This activity is intended to understand SoS top level missions and to plan a mission strategy. The essential elements considered in this activity are description of the main missions, variation points, mission location, mission risk and priority, as shown in Figure 4.4. It results on the mission functional model.

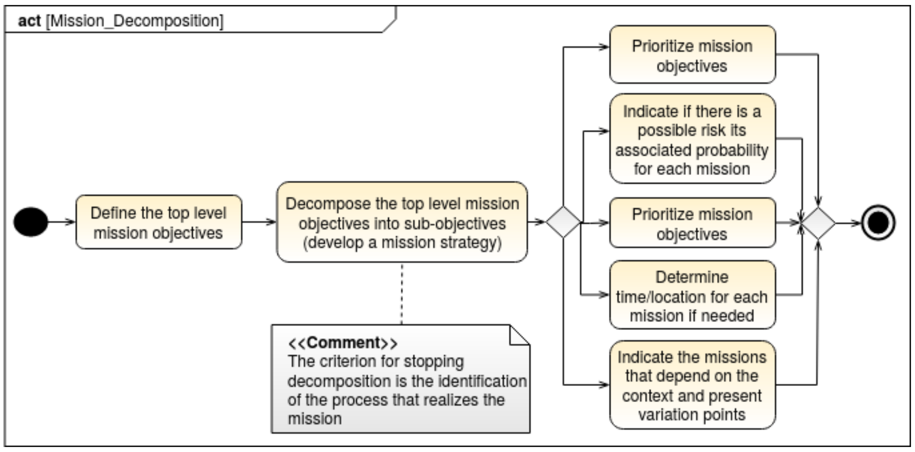

act [Mission_Decomposition]
Define the top level mission objectives
Decompose the top level mission objectives into sub-objectives (develop a mission strategy)
<<Comment>> The criterion for stopping decomposition is the identification of the process that realizes the mission
Prioritize mission objectives
Indicate if there is a possible risk its associated probability for each mission
Prioritize mission objectives
Determine time/location for each mission if needed
Indicate the missions that depend on the context and present variation points

Figure 4.4: Mission Decomposition Activity

We propose the gradual functional decomposition of the SoS mission and the splitting of complex missions into simple ones, as in goal-oriented methods (120, 121). This task is made possible by using the SysML requirement diagram (RE). The RE diagram “allows the specification of a function that a system must perform or a performance condition that a system must achieve” (122). The SysML RE diagram provides modeling constructs to represent text-based requirements and relate these requirements to other modeling elements. Different relationships are furnished to allow relating requirements to other requirements or to other model elements. These relationships include relationships for “defining a requirements hierarchy, deriving requirements, satisfying requirements, verifying requirements, and refining requirements” (122). A standard requirement includes the unique identifier and text requirement. Users can add properties if needed (122).

The basic SysML RE is not sufficient to describe all the concepts cited above. For instance, it cannot represent the mission priority and mission risk. Furthermore, it does not allow the creation of variation points since the semantic of decomposition is the conjunction. Therefore, we propose an extension of the RE diagram to allow the ADE to add the desired properties and variation points. This extension is possible since SysML is a highly extensible modeling language (122). A stereotype is one of the types of extensibility mechanisms in SysML; it is a profile class that allows designers to extend the vocabulary of SysML to create new model elements, which are derived from existing ones but have domain specific properties (122).

Figure 4.5 illustrates the extension of the SysML RE. The default properties *id* and *text* specify the unique identifier and text requirement, respectively. The *Requirement* is a stereotype that inherits from the meta-class *Class* of UML. The extension is performed by creating a stereotype called *Mission*, which contains the added properties. The stereotype Mission inherits the properties of its super-stereotype Requirement, and the following properties are added: *location, risk, priority, version, and date*.

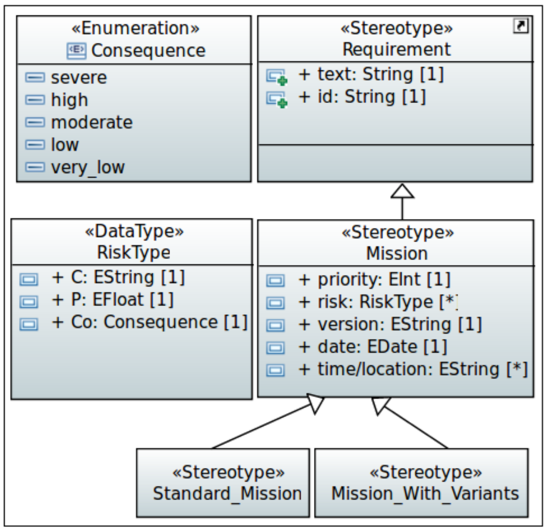


Figure 4.5: Mission Stereotype

The default property *text* of the stereotype *Requirement* can be used to describe the mission objective. The mission location may represent an IP address, GPS coordinates, polar coordinates, region, etc. For this reason, the location is considered as a string parameter. The successive refinements of the main mission generate several sub-missions. The status of these sub-missions is not the same under the main mission, and the priority is the parameter that indicates the importance of each sub-mission. We assume that the priority can take an integer value that indicates the relevance degree of the mission. Given the context uncertainty, risks can affect missions. A risk must first be identified and later mitigated if possible. Alternative activities are defined to prevent the consequences from occurring. Based on the definitions of risk found in the literature (123, 124), we propose the consideration of risk by attaching the triple R=C,P,Co to each mission (see Figure 4.5) in which C is a string value representing the future risk cause, and P is a numeric value representing the probability of risk occurrence. The suggested values for Co that represent the consequence are severe, high, moderate, low or very low.

The wave model is based on SoS upgrade cycles. Therefore the properties described above can change in each upgrade cycle. For example, the priorities are reassessed in each cycle. Therefore, we use the mission **version** as a parameter to determine the stability of the mission definition. We referred to (125) to define this parameter, in which the authors propose to use the version and date of creation/change of/in properties to indicate if and when the mission was changed.

To make our process applicable to a wide range of practical contexts, we propose supporting mission variants in the decomposition of the mission. To this end, we define two new stereotypes called the *Standard Mission* and *Mission with a Variation Point*, which inherit from the stereotype *Mission*. The *Standard Mission* is composed of a conjunction of sub-missions while the *Mission with a Variation Point* is composed of a disjunction of sub-missions.

To match the activities/actions to each mission, we propose using the *refine relationship* in the RE diagram. The *refine relationship* is used to relate a mission to another model element. We use the *call behavior action* element that references an activity to refine a mission. A mission is refined by an activity, and the referenced activity is described later using an activity diagram. Figure 5.3 illustrates part of the ***Emergency Response*** functional model. It shows the decomposition of the ***Manage Alert Stage*** *Mission with a Variation Point* into the two atomic missions ***Receive Emergency Request Description***, and ***Assess the Current Situation***, and the two composite missions ***Provide an Effective and Fast First Emergency Response***, and ***Consolidate Response***, both of them includes variation points in their definition. Each mission encapsulates the mission statement, the id, and the needed properties. Not all missions need all the properties. For example, the ***Start Emergency Response*** mission encapsu-

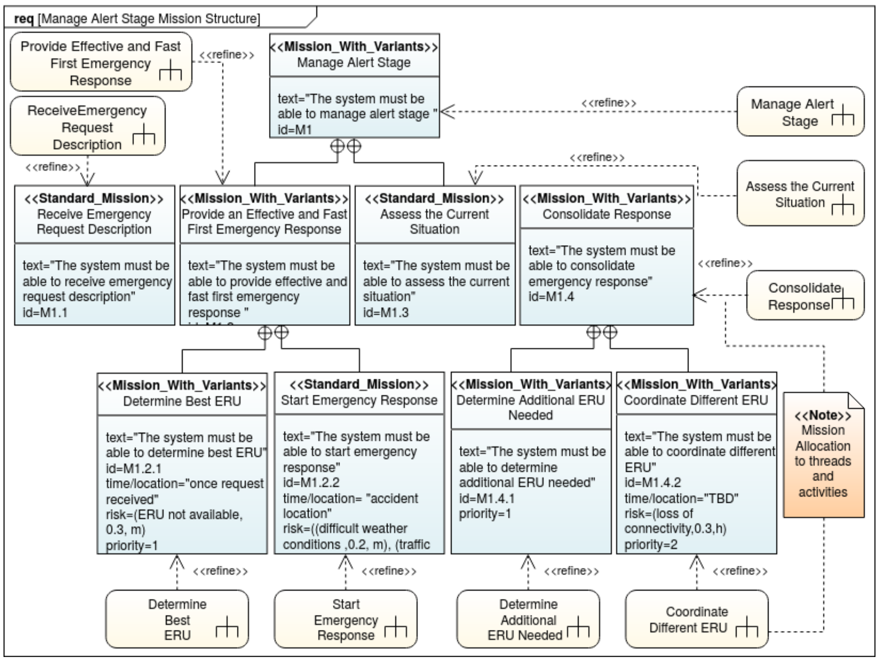

req [Manage Alert Stage Mission Structure]
Provide Effective and Fast First Emergency Response
ReceiveEmergency Request Description
<<Mission_With_Variants>> Manage Alert Stage
text="The system must be able to manage alert stage " id=M1
Manage Alert Stage
Assess the Current Situation
Consolidate Response
<<refine>>
<<Standard_Mission>> Receive Emergency Request Description
text="The system must be able to receive emergency request description" id=M1.1
<<Mission_With_Variants>> Provide an Effective and Fast First Emergency Response
text="The system must be able to provide effective and fast first emergency response "
<<Standard_Mission>> Assess the Current Situation
text="The system must be able to assess the current situation" id=M1.3
<<Mission_With_Variants>> Consolidate Response
text="The system must be able to consolidate emergency response" id=M1.4
<<Mission_With_Variants>> Determine Best ERU
text="The system must be able to determine best ERU" id=M1.2.1 time/location="once request received" risk=(ERU not available, 0.3, m) priority=1
<<Standard_Mission>> Start Emergency Response
text="The system must be able to start emergency response" id=M1.2.2 time/location= "accident location" risk=((difficult weather conditions ,0.2, m), (traffic
<<Mission_With_Variants>> Determine Additional ERU Needed
text="The system must be able to determine additional ERU needed" id=M1.4.1 priority=1
<<Mission_With_Variants>> Coordinate Different ERU
text="The system must be able to coordinate different ERU" id=M1.4.2 time/location="TBD" risk=(loss of connectivity,0.3,h) priority=2
<<Note>> Mission Allocation to threads and activities
Determine Best ERU
Start Emergency Response
Determine Additional ERU Needed
Coordinate Different ERU

Figure 4.6: Mission Decomposition and Allocation Example

lates the mission statement ***The system must be able to start emergency response***, the mission id **M.2.2**, the mission location **accident location**, and the mission risks **difficult weather conditions, and traffic jam**. Each mission, is allocated to an activity diagram as shown in Figure 5.3, the details about that is given in what follows.

4.2.2 Mission MoEs

MoEs is a traditional term widely used in systems engineering and describing how well a system carries out a task within a specific context. Sproles (126) defined MoE as "standards against which the capability of a solution to meet the needs of a problem may be judged".Thereby, in SoSE, it represents non-functional mission objectives for the SoS expressed in numerical format. It serves as the high level key performance indicators that would be automatically checked when the solution layer is specified. So, this activity focuses on the performance of the SoS's mission solution. The aim is to enhance the SoS's mission performance as much as possible.

Developing good measures can prove to be a confusing task. Thereby, guidance on this subject should be as clear, useful, and simple as possible. By carrying out a literature search on this subject, we found that the military domain attaches great importance to the development of high-quality measures to evaluate missions. Methodologies and guidelines are proposed in this field (127, 128, 129), while there is a lack of propositions in SE and SoSE domains. An interesting contribution in both SE and SoSE domains were found in (31, 130, 131), where authors proposed a framework for evaluation of architectures using MoEs. The framework identifies the elements needed to perform an evaluation and rules for acquiring or developing each of those elements so that a particular evaluation can be performed.

In our process, we relied on the works cited previously, we used the steps presented by (127), the guidelines proposed by (129), and we adapted them to SoSE domain using the ontology proposed by (31). Figure 4.7 shows the steps to follow to define MoEs and their indicators for each mission objective.

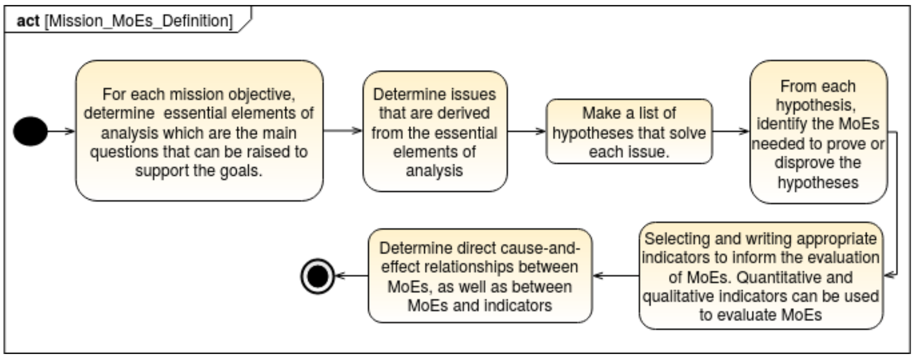

act [Mission_MoEs_Definition]
For each mission objective, determine essential elements of analysis which are the main questions that can be raised to support the goals.
Determine issues that are derived from the essential elements of analysis
Make a list of hypotheses that solve each issue.
From each hypothesis, identify the MoEs needed to prove or disprove the hypotheses
Selecting and writing appropriate indicators to inform the evaluation of MoEs. Quantitative and qualitative indicators can be used to evaluate MoEs
Determine direct cause-and-effect relationships between MoEs, as well as between MoEs and indicators

Figure 4.7: MoEs identification Activity

We can summarize these steps as follows: to adequately assess the SoS's mission effectiveness, it is needed to (1) determine essential elements of analysis, which are the main aspects that can be considered to support each mission objective. From the essential elements of analysis, (2) derive the associated issues and (3) make a list of hypotheses that solve each issue. From each hypothesis, (4) identify the MoEs needed to prove or disprove the hypotheses and (5) select the appropriate indicators to inform the evaluation of MoEs. Quantitative and qualitative indicators can be used to evaluate MoEs. It may be necessary to (6) determine direct cause-and-effect relationships between MoEs, as well as between MoEs and indicators. An example of the hierarchy of the mission objective, the elements of analysis, the issues, the hypothesis, the MoEs and the associated indicators is illustrated in Figure 4.8.

In the following table (See Table 4.1), we give an example of MoEs for the ***provide***

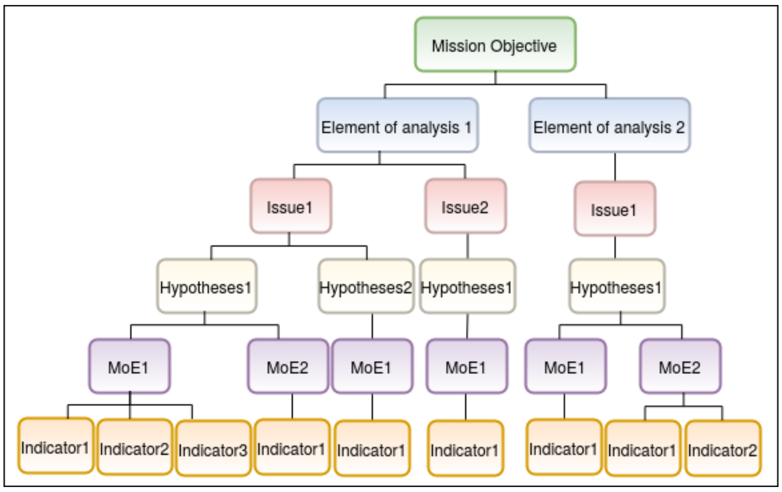


Figure 4.8: Example of the hierarchy between the mission objective, MoEs and indicators

effective and fast first emergency response mission. Two element of analysis are considered, the ***human life aspect*** and the ***environmental aspect*** of the first emergency response. In the first one, the ***first response time*** and the ***evacuation effectiveness*** are the most important MoEs, and different indicators allow to determine these MoEs like ***arrival time on scene***, ***evacuation time***, ***first survival rate***, and ***first rate of injured***.

As in the OOSEM method, we propose to capture MoEs and indicators in the SysML block definition diagram, and the methods and models for calculating the MoEs are described using the SysML parametric diagrams with the objective function as shown in Figure 4.9.

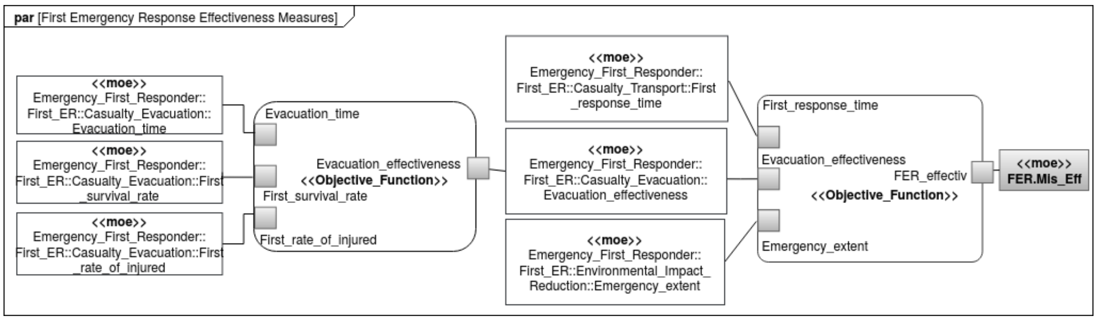

par [First Emergency Response Effectiveness Measures]
<<moe>> Emergency_First_Responder:: First_ER::Casualty_Evacuation:: Evacuation_time
<<moe>> Emergency_First_Responder:: First_ER::Casualty_Evacuation::First _survival_rate
<<moe>> Emergency_First_Responder:: First_ER::Casualty_Evacuation::First _rate_of_injured
Evacuation_time
Evacuation_effectiveness
<<Objective_Function>>
First_survival_rate
First_rate_of_injured
<<moe>> Emergency_First_Responder:: First_ER::Casualty_Transport::First _response_time
<<moe>> Emergency_First_Responder:: First_ER::Casualty_Evacuation:: Evacuation_effectiveness
<<moe>> Emergency_First_Responder:: First_ER::Environmental_Impact_ Reduction::Emergency_extent
First_response_time
Evacuation_effectiveness
FER_effectiv
<<Objective_Function>>
Emergency_extent
<<moe>> FER.Mis_Eff

Figure 4.9: First Emergency Response MoEs SysML Parametric Diagram

Table 4.1: First Emergency Response MoEs Development

Mission objective	Elements of analysis	Issues	Hypotheses	MoEs	Indicators
Provide effective and fast first emergency response	Human life impacts	Late response	The average emergency response time is seven minutes.	First response time	Arrival time on scene
		Late evacuation	Effective evacuation of injured to reach a safety place within reasonable time.	Evacuation effectiveness	Evacuation time
					First survival rate
					First rate of injured
	Environmental impacts	Large extent of impact	Reduce as much as possible the environmental impact of the emergency.	Extent of the impact	Number of cities
					Number of towns
					Square miles
					Affected population

4.2.3 Mission Definition

This phase is intended to define the activities, and course of actions of missions, considering the variability in the user environment that impacts the ways the capabilities are executed. Once the activities have been identified, they are described in this phase using the SysML activity diagram. Figure 4.10 illustrates the process to follow to define missions. The refinement principle is to (1) associate each composite mission to a mission thread represented as composed activity, and each atomic mission to an activity. Indeed, each ***call behavior action*** element identified in the mission functional model is refined using an activity diagram. Then, (2) it is important to determine for each mission thread, the contextual data of the environment that influence the course of actions, and (3) represent them using entry parameters, and refine mission threads by specifying activities, actions, parameters and alternatives. Finally, all activities have to be developed by specifying input and output parameters, actions, control flows, partitions, and constraints. The activity refinement process is stopped when the expert reaches a process composed only of actions that correspond to a capability.

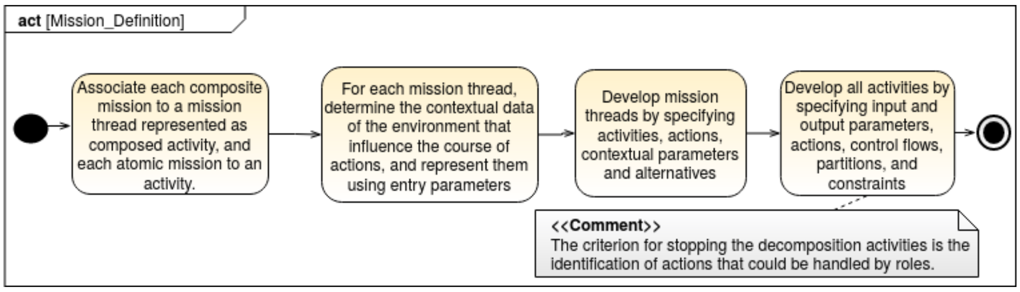

act [Mission_Definition]
Associate each composite mission to a mission thread represented as composed activity, and each atomic mission to an activity.
For each mission thread, determine the contextual data of the environment that influence the course of actions, and represent them using entry parameters
Develop mission threads by specifying activities, actions, contextual parameters and alternatives
Develop all activities by specifying input and output parameters, actions, control flows, partitions, and constraints
<<Comment>>
The criterion for stopping the decomposition activities is the identification of actions that could be handled by roles.

Figure 4.10: Mission Definition Activity

In the SysML activity diagram, ***partitions*** are used to group actions that have some common characteristics (14, 122). Partitions are commonly used to regroup actions that are performed by the same system. We propose using partitions to group all the actions that are performed by the same role, as shown in Figure 5.3, which gives an example of mission definition model that concerns the ***First Emergency Response*** mission. The mission strategy may change according to the context. We described the SoS context using activity ***entry parameters***. In this manner, the context of a mission can be inferred from the parameter values, and all the alternatives can be defined at an early stage. These parameters are always updated; for example, a location data point is always updated by a geolocation system. ***Signals*** are used to express the mission triggers (See ***Receive Order*** signal in Figure 5.3), while ***action*** scheduling can be expressed using *activities, actions, data and control flows*. ***Constraints*** can be set on actions to specify the business semantics. The mission definition phase results in the fine grained behavioral view of the sub-missions, called ***the mission definition model***, which is defined using the SysML activity diagram (See Figure 5.3).

4.2.4 Role Definition and Assignment

The role definition phase focuses on constituent system level information that impacts the SoS mission. The roles, capabilities and the possible constituent systems are ex-

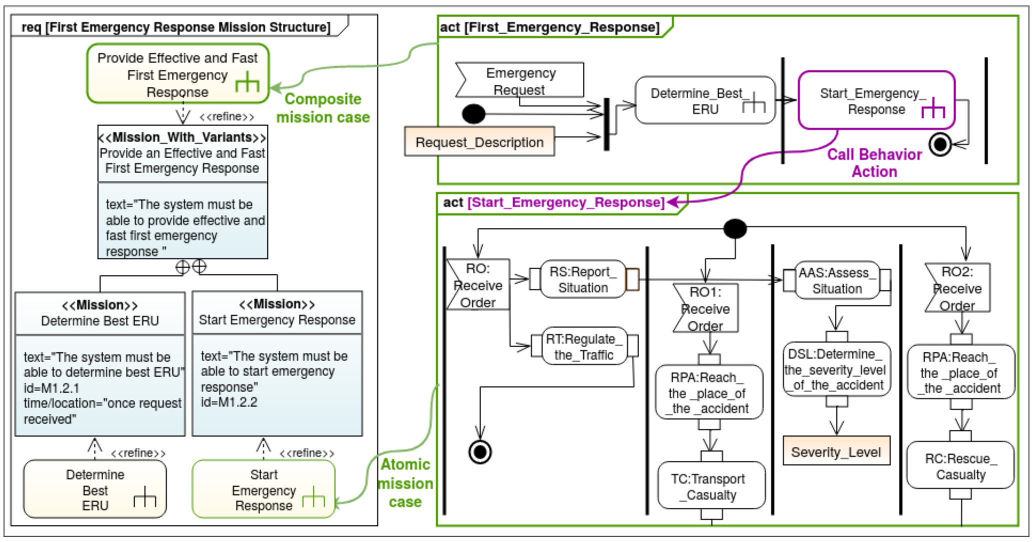

req [First Emergency Response Mission Structure]
Provide Effective and Fast First Emergency Response
Composite mission case
<<refine>>
<<Mission_With_Variants>> Provide an Effective and Fast First Emergency Response
text="The system must be able to provide effective and fast first emergency response "
<<Mission>> Determine Best ERU
text="The system must be able to determine best ERU" id=M1.2.1 time/location="once request received"
<<Mission>> Start Emergency Response
text="The system must be able to start emergency response" id=M1.2.2
Determine Best ERU
Start Emergency Response
Atomic mission case
act [First_Emergency_Response]
Emergency Request
Request_Description
Determine_Best_ERU
Start_Emergency_Response
Call Behavior Action
act [Start_Emergency_Response]
RO: Receive Order
RS:Report_Situation
RT:Regulate_the_Traffic
RO1: Receive Order
RPA:Reach_the_place_of_the_accident
TC:Transport_Casualty
AAS:Assess_Situation
DSL:Determine_the_severity_level_of_the_accident
Severity_Level
RO2: Receive Order
RPA:Reach_the_place_of_the_accident
RC:Rescue_Casualty

Figure 4.11: Mission Definition Example

plored. Given the uncertainty of SoS boundaries, the challenge is to include the roles that can support the SoS mission, and to exclude the useless role ones. In our approach, the ADE considers roles that she/he judges to be relevant entities to the SoS mission. A constituent system enters the SoS boundary when it begins to affect the SoS behavior and leaves when its contribution is negated (41).

Role modeling can be performed using a block definition diagram, in which the structural and behavioral features of a role are described (See Figure 4.12). Hierarchical relationships between the roles can be defined. The capabilities are modeled using operations. Coherent set of capabilities are grouped into interfaces. A *realization dependency* is added from the role to each provided interface, which means that the role will provide each capability in that interface. A role can assert that it requires a set of capabilities by adding a uses dependency to an interface. We created a stereotype Role that inherits the properties of its superstereotype *block* to use SoS vocabulary domain. The communication between roles is done by sub-typing their communication media. For example information exchange could be done by the use of Radio communication.

The purpose of the role assignment phase is to allow the expert to determine the constituent systems that will fulfill actions. Therefore, she/he provides a model that combines the functional and structural views of the mission. As shown in Figure 4.13, role assignment is based on the functional allocation of activity partition into a role. The role must have capabilities that allow it to achieve all actions contained in the

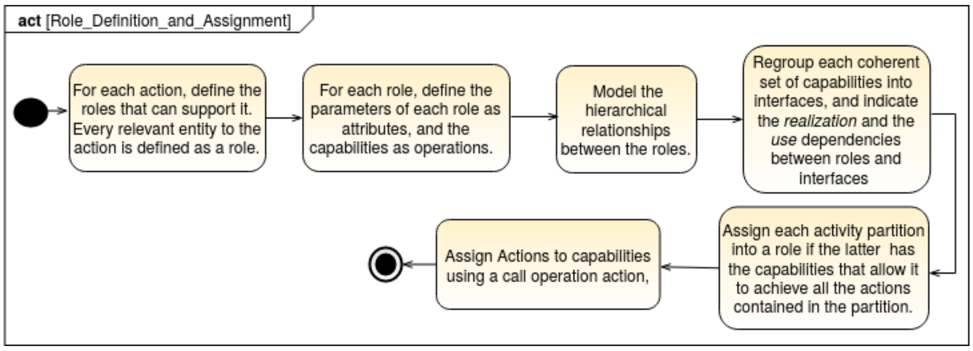

act [Role_Definition_and_Assignment]
For each action, define the roles that can support it. Every relevant entity to the action is defined as a role.
For each role, define the parameters of each role as attributes, and the capabilities as operations.
Model the hierarchical relationships between the roles.
Regroup each coherent set of capabilities into interfaces, and indicate the realization and the use dependencies between roles and interfaces
Assign each activity partition into a role if the latter has the capabilities that allow it to achieve all the actions contained in the partition.
Assign Actions to capabilities using a call operation action,

Figure 4.12: Role Definition and Assignment Activity

partition. Actions are allocated to capabilities using a call operation action, as shown in Figure 4.13.

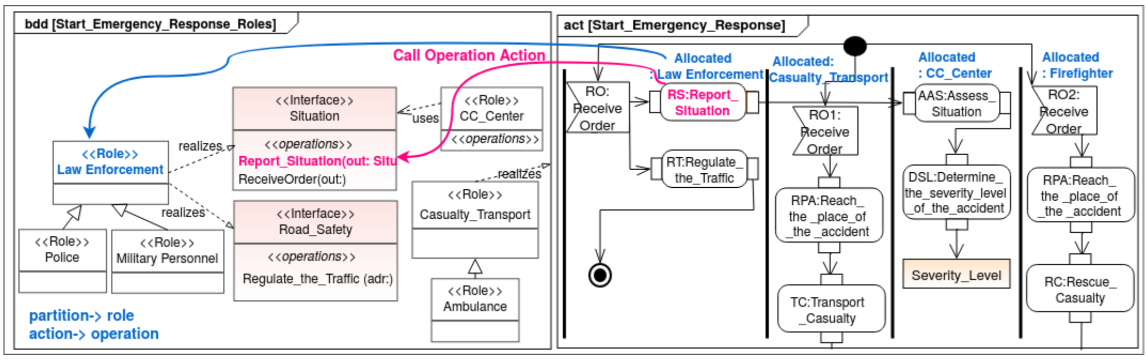


Figure 4.13: Role Definition and Assignement Example

4.2.5 Abstract Architectures

The next step in our approach is to generate the SoS abstract architecture. The focus of this phase is describing which abstract roles interact within a configuration and how. We propose automatizing as much as possible the process of generating the abstract architecture, from the activity diagram and the roles's BDD, using ATL (ATLAS Transformation Language) [1]. This abstract architecture is defined using the SysML Block Definition Diagram (BDD) and Internal Block Diagram (IBD). The BDD defines logical decomposition of the main block into roles, and the IBD defines the interactions

[1]Model transformation language and toolkit. Web site: https://www.eclipse.org/atl/ Accessed 19/07/2021

among the roles such that they satisfy the mission (see Figure 4.14). We used the mission structural and behavioral model as source model, and the target model is the BDD and the IBD diagrams. The mapping between the source model and the target model is shown is illustrated in the Table 4.2. In the BDD, a block is created for each decomposition of the partition hierarchy. The block features are captured from the role BDD.

The IBD captures the internal structure of a block in terms of the parts, properties and connectors, and this structure is used to display different connections between the parts (roles) that compose the block. The main idea of the transformation here is to consider the activity diagram as a starting point. Considering that an activity refines a mission, our main goal is to build the block that can satisfy this mission (see Figure 4.14). To this end, we associate each activity model element to a block element that has the name of the model; this block is considered as the main block.

Table 4.2: Mission structural and behavioral model- Abstract Architecture correspondence

Mission structural and behavioral model element	**Abstract architecture element**
Activity (Call behavior action)	Composite block
Activity input parameter (resp. output)	Port on the block that corresponds to the activity that needs (resp. provide) the parameter
Role (partition)	Block
Action (call operation action)	Operation in the block that corresponds to the role that handles the action
Signal	Signal in the block that corresponds to the role that send or receive the signal
Pin	Port on the block that corresponds to the role that contains the pin
Flot	Connector
Data store	Attribute
Constraint	Constraint

Since our activity diagram is composed mainly of partitions allocated to roles, the partitions are represented as parts in the main block. The parts have the same names and types as the partitions. An activity is characterized by input (or output) parameters. The parameters are provided (or required) by other blocks, which means that each parameter corresponds to an interaction point that is represented in the IBD by a block port. Each flow between the actions from different partitions indicates that data exchange occurs between two different roles. This aspect means that a flow between the two corresponding parts must be created using the corresponding ports. The elements of the resulting IBD are allocated since they are generated from the activity diagram. The resulting IBD is consistent with the activity diagram, as shown in Figure 4.14.

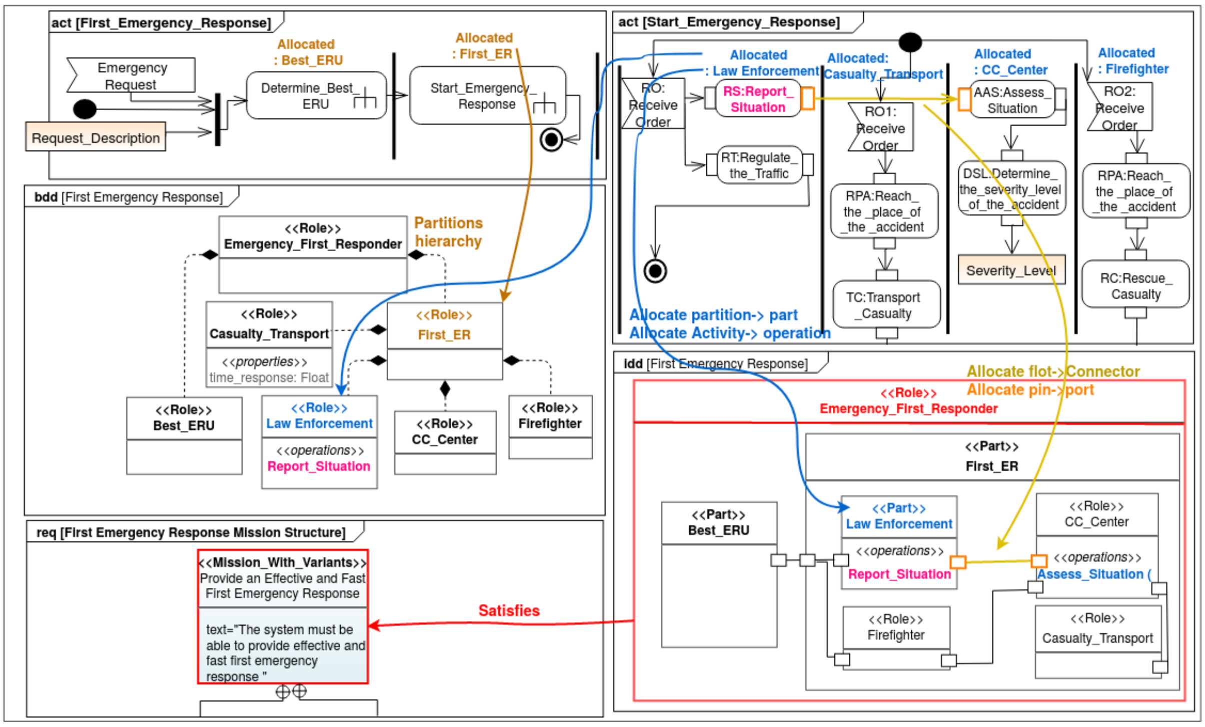
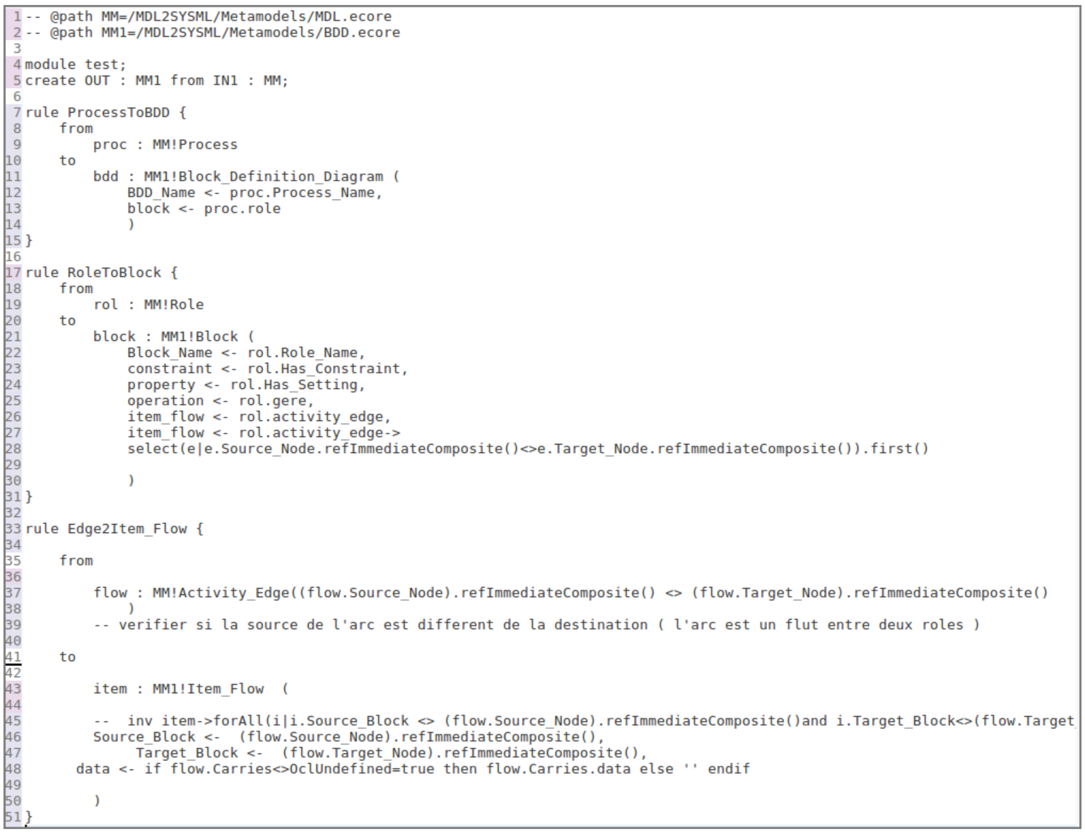

act [First_Emergency_Response]
Emergency Request
Request_Description
Allocated : Best_ERU
Determine_Best_ERU
Allocated : First_ER
Start_Emergency_Response
bdd [First Emergency Response]
<<Role>> Emergency_First_Responder
Partitions hierarchy
<<Role>> Casualty_Transport
<<properties>> time_response: Float
<<Role>> First_ER
<<Role>> Best_ERU
<<Role>> Law Enforcement
<<operations>> Report_Situation
<<Role>> CC_Center
<<Role>> Firefighter
req [First Emergency Response Mission Structure]
<<Mission_With_Variants>> Provide an Effective and Fast First Emergency Response
text="The system must be able to provide effective and fast first emergency response "
Satisfies
act [Start_Emergency_Response]
Allocated : Law Enforcement
Allocated: Casualty_Transport
Allocated : CC_Center
Allocated : Firefighter
RO: Receive Order
RS:Report_Situation
RT:Regulate_the_Traffic
RO1: Receive Order
RPA:Reach_the_place_of_the_accident
TC:Transport_Casualty
AAS:Assess_Situation
DSL:Determine_the_severity_level_of_the_accident
Severity_Level
RO2: Receive Order
RPA:Reach_the_place_of_the_accident
RC:Rescue_Casualty
Allocate partition-> part
Allocate Activity-> operation
ibd [First Emergency Response]
Allocate flot->Connector
Allocate pin->port
<<Role>> Emergency_First_Responder
<<Part>> First_ER
<<Part>> Best_ERU
<<Part>> Law Enforcement
<<operations>> Report_Situation
<<Role>> CC_Center
<<operations>> Assess_Situation (
<<Role>> Firefighter
<<Role>> Casualty_Transport

Figure 4.14: Abstract Architecture Example

To implement the transformation, we used simplified meta-models of the activity, BDD and IBD diagrams of SysML. We defined them using Eclipse Core (ECore) [1]. Figure 4.15 depicts part of the ATL transformation rules. To access the elements of a model, ATL uses query to navigate between the elements of a model and to call operations on them.

[1]Common standard for expressing models . Web site: https://www.eclipse.org/ecoretools/ Accessed 03/08/2021



```
-- @path MM=/MDL2SYSML/Metamodels/MDL.ecore
-- @path MM1=/MDL2SYSML/Metamodels/BDD.ecore

module test;
create OUT : MM1 from IN1 : MM;

rule ProcessToBDD {
    from
        proc : MM!Process
    to
        bdd : MM1!Block_Definition_Diagram (
            BDD_Name <- proc.Process_Name,
            block <- proc.role
            )
}

rule RoleToBlock {
    from
        rol : MM!Role
    to
        block : MM1!Block (
            Block_Name <- rol.Role_Name,
            constraint <- rol.Has_Constraint,
            property <- rol.Has_Setting,
            operation <- rol.gere,
            item_flow <- rol.activity_edge,
            item_flow <- rol.activity_edge->
            select(e|e.Source_Node.refImmediateComposite()<>e.Target_Node.refImmediateComposite()).first()

            )
}

rule Edge2Item_Flow {

    from

        flow : MM!Activity_Edge((flow.Source_Node).refImmediateComposite() <> (flow.Target_Node).refImmediateComposite()
            )
        -- verifier si la source de l'arc est different de la destination ( l'arc est un flut entre deux roles )

    to

        item : MM1!Item_Flow  (

        --  inv item->forAll(i|i.Source_Block <> (flow.Source_Node).refImmediateComposite()and i.Target_Block<>(flow.Target
        Source_Block <-  (flow.Source_Node).refImmediateComposite(),
             Target_Block <-  (flow.Target_Node).refImmediateComposite(),
      data <- if flow.Carries<>OclUndefined=true then flow.Carries.data else '' endif

        )
}
```

Figure 4.15: ATL Transformation Rules

4.2.6 Capture measures MoPs

This phase focuses on the performance of the SoS solution. The aim is to enhance the SoS's mission performance as much as possible. Measures of Performance (MoPs) "provide an assessment from an engineering perspective of a particular solution to a given specification, for instance a MoP can be applied to a specific service to verify it against its specification to validate that it performs correctly" (31, 132). MoEs are based on MoPs, which are carried out for each service in the architecture. Because the primary goal of MoPs is to determine whether a service meets its specifications, as shown in Figure 4.16, MoPs only use the service specification and not a scenario.

Figure 4.16 depicts the relationship between MoEs and MoPs in more details, MoEs are seen as needing MoPs as building blocks to complete the MoE assessment. Therefore MoPs can only be used to validate a service against its specification, whilst MoEs allow assessments to be made about a capability's ability to satisfy a sce-

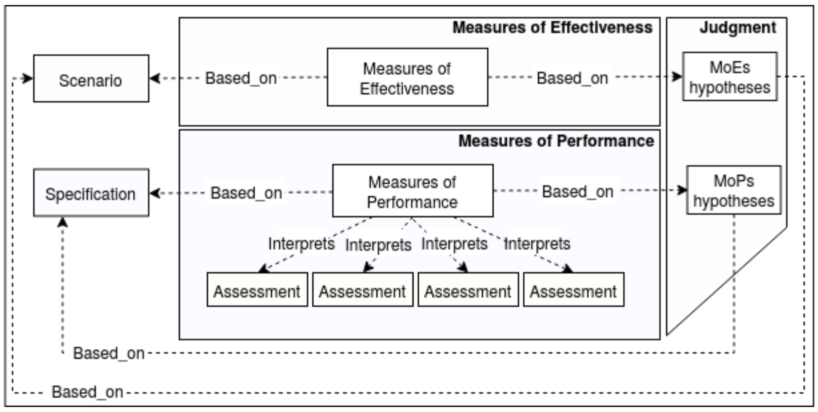


Figure 4.16: Relationship of MoEs with MoPsm (31)

nario (31). In context of SoSs, a service can be offered by several constituent systems, in fact, we used the MoPs to choose between several candidate constituent systems that can concertize a role. Several criteria can be considered, namely, the availability, cost, performance, etc.

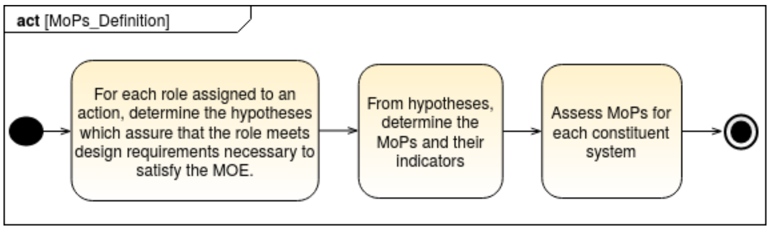

act [MoPs_Definition]
For each role assigned to an action, determine the hypotheses which assure that the role meets design requirements necessary to satisfy the MOE.
From hypotheses, determine the MoPs and their indicators
Assess MoPs for each constituent system

Figure 4.17: Capture MoPs Activity

In Figure 4.17, the steps to follow to define MoPs are shown. The system architect has to (1) determine the hypotheses which ensure that the role meets design requirements necessary to satisfy the MOEs of the action. Then, she/he (2) deduces the MoPs and the related indicators that ensure the role has the capability to perform. Finally, MoPs for each constituent system are performed to support the choice between alternative solutions. As for MoEs, MoPs are captured using the SysML parametric diagram.

Figure 4.18 illustrates an example of MoPs for the ambulance (moving mean). Availability, travel cost and efficiency are the main MoPs for the ambulance. The efficiency of an ambulance is calculated using the mobilization time (from the moment an emergency call is placed in queue for an ambulance to the time the first vehicle is en route to the incident), travel time (the time an ambulance takes to drive to the scene

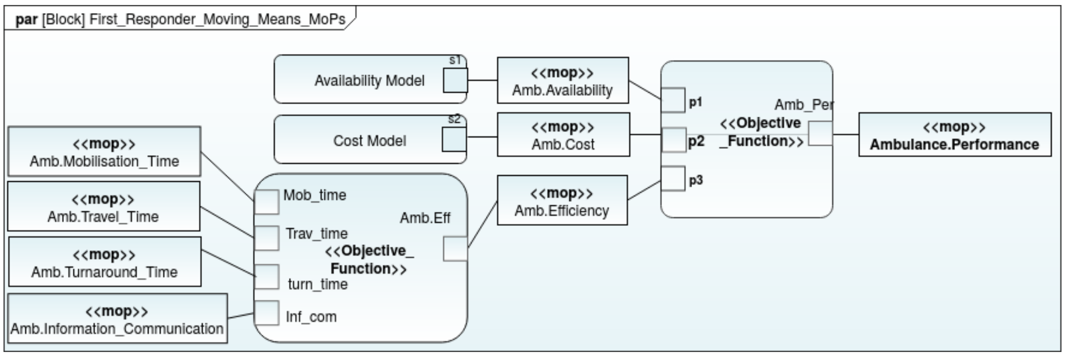


Figure 4.18: First Responder Moving Means MoPs

of an incident), turnaround time (from the moment an ambulance which is transporting a patient arrives at a hospital to when the ambulance is ready to respond to a new incident.), and information communication (ability to communicate informations).

4.2.7 Concrete systems requirements and concrete architecture

The abstract architecture is used to obtain one of the possible concrete architectures. To define the concrete architecture that will satisfy the mission, we used the steps defined in the Figure 4.19. We define the concrete architecture in terms of its physical constituent systems, their relationships, and their distribution across the SoS nodes. The physical constituent systems of the SoS are represented by hardware systems, software systems, humans, organizations, and the data exchanged. As shown in Figure 4.19, the performance measures are used to select the preferred constituent system. Furthermore, the CSs interoperate with each other (communicate, exchange data, etc), and each CS uses a set of interoperability mechanisms (Radio communication, ADSL connection, etc). Therefore, interoperability is possible by sub-typing the communication media or by indirection via another CS. For instance, the control and command center can obtain data from the ambulance via an Radio communication. We emphasize that the data and their nature have to be specified in the concrete architecture.

At CSs level, the SoS architect can identify new requirements needed at SoSs. Theses requirements can represent a new feature, resource, or means of interoperability, needed to improve the mission achievement. The CSs new requirements require negotiation with CSs engineers, and are described using the SysML requirement diagram.

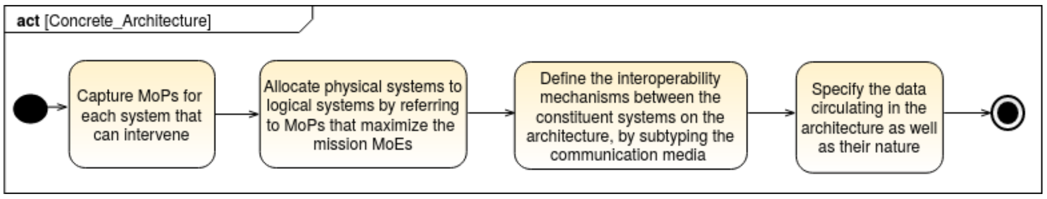

act [Concrete_Architecture]
Capture MoPs for each system that can intervene
Allocate physical systems to logical systems by referring to MoPs that maximize the mission MoEs
Define the interoperability mechanisms between the constituent systems on the architecture, by subtyping the communication media
Specify the data circulating in the architecture as well as their nature

Figure 4.19: Define Concrete Architecture Activity

4.3 Discussion

The main goal of this work is to consider mission thread as SOI (System Of Interest) to bridge the dissociation between SoS objectives, and the individual functionalities undertaken by constituent systems, to support the SoS mission. To address this SoSE challenge, we proposed along this chapter to maintain a mission focus throughout the SoSE analysis and the architecting process included in the wave model. In the following, we discuss our proposition while returning to SoSs characteristics and to the state of the art.

4.3.0.1 Back to SoSs characteristics

This section returns to a subset of challenge problems that which derive from the characteristics of SoSs, and considers how each one is addressed based on the models that have been developed in the illustrative example.

- Operational independence: which means that any constituent system is independent and can operate serviceably if the SoS is disassembled. In the proposed process, we considered that a constituent system is an independent entity that is able to provide to the SoS a subset of its functionalities, which are called capabilities. The mission actions are allocated to roles and the roles are replaced by available constituent systems based on performance measures. Thus, when a constituent system is disassembled from the SoS, it continues to operate independently.

- Dynamic environment: constituent systems supporting each role in a mission will vary over the course of the actions. The mission context is key player to determine the constituent systems that are involved in mission accomplishment. We use set of parameters to determine the mission context, when parameters change, the course of actions in the activity diagram changes. The activity diagram in Figure 5.3 shows a decision node that depends on the security event state. The activity that will be executed depends on the value of risk. Indeed, different architectures are generated for different risk values.

- SoS evolution: SoS evolves over time, but evolves slowly. The evolution could consist of the addition of a new constituent system or a change in the behavior of a constituent system. In the case of addition, we add the new constituent system in the role diagram and add the generalization relationships towards the appropriate roles, or create a new role if the new constituent system holds a new behavior. In the case of a change in the behavior of a constituent system, the role diagram is also updated according to the new capabilities. In the two cases (addition or modification of constituent system), the measures of performance are made and compared to the MoP of the constituent systems holding the same capabilities. The challenge here for the architect is to propagate changes across the concrete architecture.

4.3.0.2 Back to the state of the art

SE approaches are mature and guide engineers in the analysis, development and documentation of complex systems. The majority of them are based on SysML to take advantage of its syntactic richness. We took advantage of their maturity from the methodological point of view: we captured performance and effectiveness measures using the parametric diagram as in OOSEM. For the role definition phase, we used the service-based interactions provided and required as Harmony SE does to describe the system components. We have refined the RE diagram directly by an activity diagram as in MagicGrid. However, while these approaches are based on the concept of *requirement* that must be met, our approach is based on the *mission* paradigm. Through this paradigm we want to offer a more operational view that supports all the tactical information that can change the order of execution of the actions, the involved systems and different operational environments. Thus, we considered in the decomposition through the RE diagram the different points of variation of a mission, something that is not considered in the SE approaches. These points of variation are solved by activity diagrams taking contextual information as input parameters to show the different alternatives and decisions. So refinement does not just consider leaf missions as in other approaches.

SoSE work is most often influenced by DoDAF and MoDAF frameworks. For example, the Compass and Danse projects have tried to reduce the number of views, proposed by these frameworks, to make them more manageable. An example of these influences concerns the fact of considering concrete systems during the specification stage. So, SoS boundaries are dictated by existing systems. In our approach, the SoS specification is more open thanks to the use of a more abstract definition of constituents (Role concept).

Chapter 5

CASE STUDY MODELING AND SIMULATION

In this chapter, we discuss and explain the steps of the proposed process using a case study. We focus on SoS dedicated to manage crowd during a football event. First, we briefly outline the case study. Then, in order to illustrate our approach, we detail how a crowd management SoS is specified in our approach. Finally, in the last section, we provide an evaluation and discussion of the case study.

5.1 Case study presentation

The crowd management SoS case study was defined in (4). It aimed at developing an integrated crowd control system, during temporary events, of mass transit, such as sports events or political meetings. Thus, crowd management can be done at various locations, such as conference rooms, streets, airports, train stations, stadiums, malls and many more. In this chapter, we focus on the management of crowd in sport events. In this case, the SoS could involve several constituent systems, as shown in Figure 5.1, such as organizations (Command and control center, fire brigade, police, catering facilities, transportation and transit committee, etc.), humans (Stewards, decision-makers, firefighters, policeman, transport operators, etc.), and technical systems (Sensors, drones, communication means, etc.).

The main objective of the case study is to design an SoS, which responds to the crowd management mission, and respect the overall mission effectiveness (Response time to an emergency, crowd satisfaction criteria, crowd control criteria, etc.). The SoS must observe and monitor the crowd, and act directly on risk and incidents. For this, the SoS must be designed to ensure relevant information flow between CSs, and the cooperation among the various autonomous CSs, beyond the boundary of individual organizations.

The choice of this case study was motivated by the fact, that the crowd management application domain still remain widely open, due to highly complex behaviors driven by individuals. Thus, Several issues with crowd modeling are open to date. We cite as example the modeling of the behavior, and the cooperation of the involved CSs, to provide more effective response to the crowd management mission (133, 134).

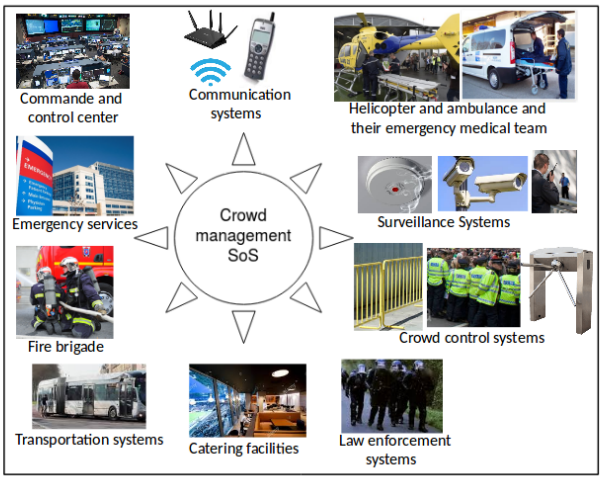


Figure 5.1: Crowd Management Case Study

The case study data were collected primarily using document analysis. We tried to use several sources of information, as suggested by (135), in order to limit the effects of one interpretation of one single data source. For data related to crowd management in football events venues, we mainly relied on the FIFA (Fédération Internationale de Football Association) [1] regulations (136, 137), and other sources (134, 138, 139, 140, 141). To establish mission threads, we used also several guides for emergency plan development (142, 143, 144, 145).

[1]https://www.fifa.com/ Accessed 13/01/2021

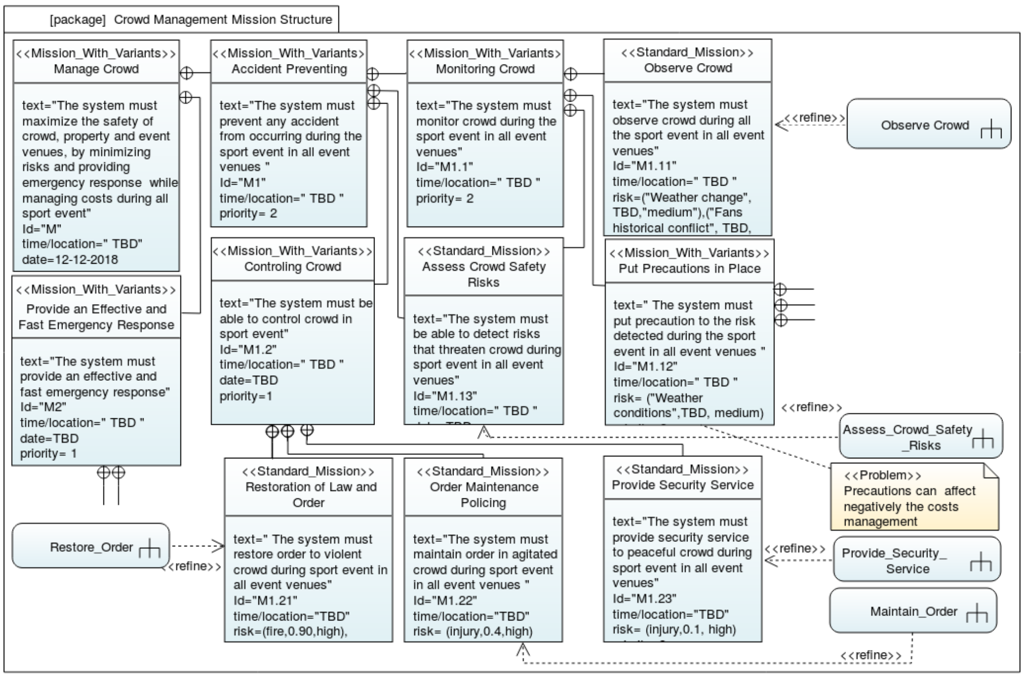

[package] Crowd Management Mission Structure
<<Mission_With_Variants>> Manage Crowd
text="The system must maximize the safety of crowd, property and event venues, by minimizing risks and providing emergency response while managing costs during all sport event" Id="M" time/location=" TBD" date=12-12-2018
<<Mission_With_Variants>> Accident Preventing
text="The system must prevent any accident from occurring during the sport event in all event venues " Id="M1" time/location=" TBD " priority= 2
<<Mission_With_Variants>> Monitoring Crowd
text="The system must monitor crowd during the sport event in all event venues" Id="M1.1" time/location=" TBD " priority= 2
<<Standard_Mission>> Observe Crowd
text="The system must observe crowd during all the sport event in all event venues" Id="M1.11" time/location=" TBD " risk=("Weather change", TBD,"medium"),("Fans historical conflict", TBD,
<<refine>>
Observe Crowd
<<Mission_With_Variants>> Provide an Effective and Fast Emergency Response
text="The system must provide an effective and fast emergency response" Id="M2" time/location=" TBD " date=TBD priority= 1
<<Mission_With_Variants>> Controling Crowd
text="The system must be able to control crowd in sport event" Id="M1.2" time/location=" TBD " date=TBD priority=1
<<Standard_Mission>> Assess Crowd Safety Risks
text="The system must be able to detect risks that threaten crowd during sport event in all event venues" Id="M1.13" time/location=" TBD "
<<Mission_With_Variants>> Put Precautions in Place
text=" The system must put precaution to the risk detected during the sport event in all event venues " Id="M1.12" time/location=" TBD " risk= ("Weather conditions",TBD, medium)
<<refine>>
Assess_Crowd_Safety _Risks
<<Problem>> Precautions can affect negatively the costs management
<<Standard_Mission>> Restoration of Law and Order
text=" The system must restore order to violent crowd during sport event in all event venues" Id="M1.21" time/location="TBD" risk=(fire,0.90,high),
<<Standard_Mission>> Order Maintenance Policing
text="The system must maintain order in agitated crowd during sport event in all event venues " Id="M1.22" time/location="TBD" risk= (injury,0.4,high)
<<Standard_Mission>> Provide Security Service
text="The system must provide security service to peaceful crowd during sport event in all event venues" Id="M1.23" time/location="TBD" risk= (injury,0.1, high)
Restore_Order
<<refine>>
<<refine>>
Provide_Security_ Service
Maintain_Order
<<refine>>

Figure 5.2: Mission Decomposition Diagram

5.1.1 Mission decomposition

The initial top level mission for the crowd management SoS is to *maximize the safety of the crowd, property and event venues, by minimizing risks and providing emergency response while managing costs during all sports events*. This mission is decomposed into several sub-missions. The decomposition is a part of the mission structure package. Figure 5.2 shows part of the realized mission functional model. The initial mission is represented by the *manage crowd* mission, which includes the text statement describing the mission and the "M" id. Two second level sub-missions are designed, namely, *accident prevention* and *provide an effective and fast emergency response*, and *accident prevention* has a lower priority than the second one. The *crowd management* mission is a mission with variants. The semantic of decomposition is the OR operator since the *provide an effective and fast emergency response* mission is not executed if there is no emergency request. The mission time/location are not determined yet and depend on the event location. Thus, we use the TBD (to be determined) acronym to indicate that the values will be determined later. The decomposition process is iterative. For instance, the accident prevention sub-mission is decomposed into crowd control and crowd monitoring sub-missions, and the crowd monitoring sub-mission is further decomposed into several sub-missions. The decomposition is stopped when we can refine a sub-mission by an activity using the refine relationship. Using the SysML refine relationship allows the reuse of the SysML traceability mechanisms. The observe crowd call behavior action is used to refine the observe crowd sub-mission by adding a set of activity description. Each activity encompasses a set of activities/actions that refine the sub-mission and take into account the corresponding risks.

5.1.2 Mission Definition

For each *call behavior action* refining a sub-mission in the Mission Decomposition Diagram, an activity is created with the same name. This aspect ensures that each mission is associated with a set of ordered actions. Figure 5.3 shows the *Put Precautions in Place* activity allocated to the mission *Put Precautions in Place* using the *refine* relationship. In the activity diagram, the entry parameter *Security_event_state* is necessary to choose the appropriate action to be undertaken by role.

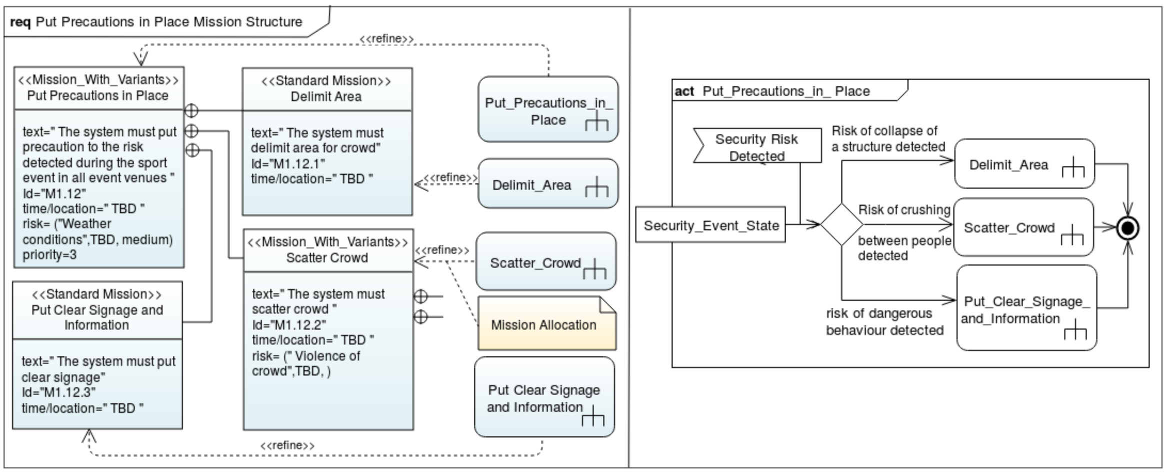

req Put Precautions in Place Mission Structure
<<refine>>
<<Mission_With_Variants>> Put Precautions in Place
text=" The system must put precaution to the risk detected during the sport event in all event venues " Id="M1.12" time/location=" TBD " risk= ("Weather conditions",TBD, medium) priority=3
<<Standard Mission>> Delimit Area
text=" The system must delimit area for crowd" Id="M1.12.1" time/location=" TBD "
Put_Precautions_in_Place
Delimit_Area
<<Mission_With_Variants>> Scatter Crowd
text=" The system must scatter crowd " Id="M1.12.2" time/location=" TBD " risk= (" Violence of crowd",TBD,)
Scatter_Crowd
Mission Allocation
<<Standard Mission>> Put Clear Signage and Information
text=" The system must put clear signage" Id="M1.12.3" time/location=" TBD "
Put Clear Signage and Information
act Put_Precautions_in_ Place
Security Risk Detected
Security_Event_State
Risk of collapse of a structure detected
Risk of crushing between people detected
risk of dangerous behaviour detected
Delimit_Area
Scatter_Crowd
Put_Clear_Signage_and_Information

Figure 5.3: Mission Decomposition and Allocation Example

The activity *Observe Crowd* (See Figure 5.4) refines the *Observe Crowd* mission. The activity that realizes the *Observe Crowd* actions when a high risk of conflict is identified is based on observation actions inside and outside the stadium. Figure 5.4 shows the activity that realizes the *Observe Crowd in Stadium* actions. Each action is stored in a partition block, and the partitions are allocated to the corresponding roles.

Observe Entry Points, Observe Troublemakers or *Observe Spectators* are examples of observation actions that provide observations as outputs. The *analyze observation* activity retrieves the observations and generates a state of safety and security in the stadium. The *analyze observation* activity is represented by *call behavior action* (see Figure 5.4), which means that it is associated with an activity diagram that describes it.

The observation actions should take into account several factors such as political tensions, historical rivalry between fans, and supporter profiles. Each contextual information is given as an input by the activity parameter (see Figure 5.4). The expected result from the observe crowd with historical conflict between fans activity is the secure event state that is provided as an output parameter.

Mission Decomposition Diagram is not supposed to contain partitions allocations. To avoid presenting two similar diagrams, one with the allocations and the second with no allocations, we have presented only Mission Decomposition Diagram with allocations.

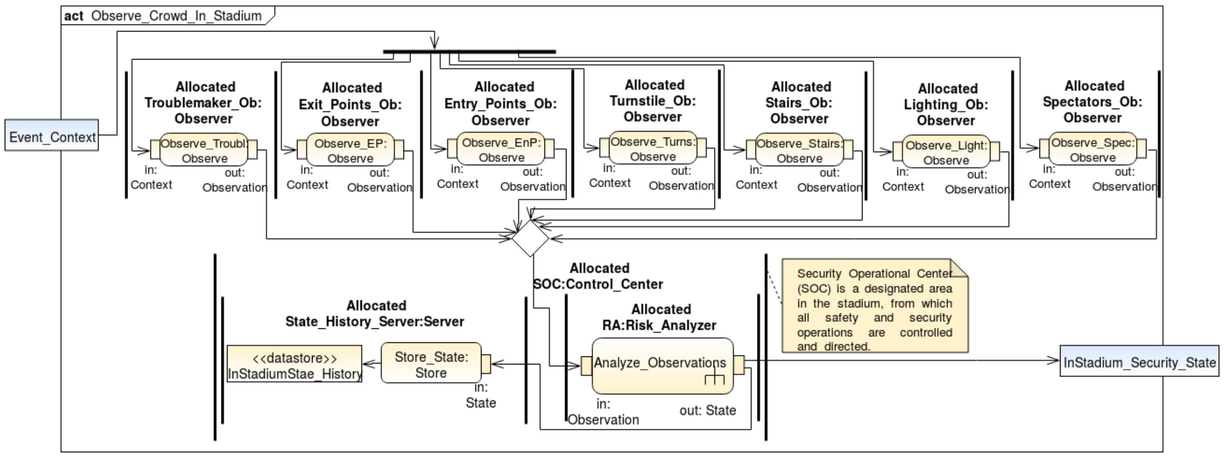

act Observe_Crowd_In_Stadium
Event_Context
Allocated Troublemaker_Ob: Observer
Observe_Troubl: Observe
in: Context
out: Observation
Allocated Exit_Points_Ob: Observer
Observe_EP: Observe
in: Context
out: Observation
Allocated Entry_Points_Ob: Observer
Observe_EnP: Observe
in: Context
out: Observation
Allocated Turnstile_Ob: Observer
Observe_Turns: Observe
in: Context
out: Observation
Allocated Stairs_Ob: Observer
Observe_Stairs: Observe
in: Context
out: Observation
Allocated Lighting_Ob: Observer
Observe_Light: Observe
in: Context
out: Observation
Allocated Spectators_Ob: Observer
Observe_Spec: Observe
in: Context
out: Observation
Allocated SOC:Control_Center
Allocated State_History_Server:Server
<<datastore>> InStadiumStae_History
Store_State: Store
in: State
Allocated RA:Risk_Analyzer
Analyze_Observations
in: Observation
out: State
Security Operational Center (SOC) is a designated area in the stadium, from which all safety and security operations are controlled and directed.
InStadium_Security_State

Figure 5.4: Mission Definition Diagram

5.1.3 Role Definition and Assignment

Once actions/activities have been defined, they must be attributed to a role. A role can be abstract to varying degrees and can be specialized using the inheritance relationship. Figure 5.5 shows part of the crowd management Role BDD, which includes a set of role hierarchies from the most abstract role to concrete roles. For instance, according to the situation, cost, service availability, and observation target, the *observation* action could be performed using a *camera*, *steward*, or *smoke detector*, etc.

As shown in Figure 5.5, the role observer provides a capability *Crowd Observation*. *Crowd observation* is required by the risk analyzer. The assignment of roles to actions is performed by typing partitions with roles. For instance, in Figure 5.4, the safety equipment observer partition is typed by the observer role, because it is not possible to determine at this stage if a camera will exist in this place or another physical entity will be used.

The observer role provides a capability *Crowd Observation*, which includes an operation called *observe*, as shown in Figure 5.5. A call operation action for *observe* is shown with pins corresponding to the entry and output parameters with all observation actions; for instance, *Observe Troublemakers* and *Observe spectators* call the operation observe, as shown in Figure 5.4. Allocating roles to partitions allows the maintenance of traceability between the role base and the activities/actions related to a sub-mission.

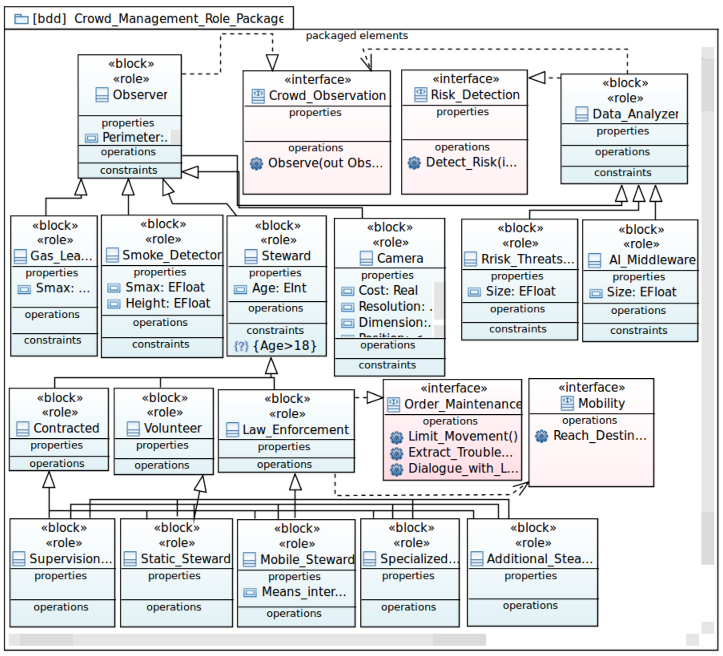

[bdd] Crowd_Management_Role_Package
packaged elements
«block» «role» Observer
Perimeter:.
«interface» Crowd_Observation
Observe(out Obs...
«interface» Risk_Detection
Detect_Risk(i...
«block» «role» Data_Analyzer
«block» «role» Gas_Lea...
Smax: ...
«block» «role» Smoke_Detector
Smax: EFloat
Height: EFloat
«block» «role» Steward
Age: EInt
{Age>18}
«block» «role» Camera
Cost: Real
Resolution: .
Dimension:.
«block» «role» Rrisk_Threats...
Size: EFloat
«block» «role» AI_Middleware
Size: EFloat
«block» «role» Contracted
«block» «role» Volunteer
«block» «role» Law_Enforcement
«interface» Order_Maintenance
Limit_Movement()
Extract_Trouble...
Dialogue_with_L...
«interface» Mobility
Reach_Destin...
«block» «role» Supervision...
«block» «role» Static_Steward
«block» «role» Mobile_Steward
Means_inter...
«block» «role» Specialized...
«block» «role» Additional_Stea...
properties
operations
constraints

Figure 5.5: BDD Roles Diagram

Figure 5.6 shows the mission-level performance and effectiveness measures that are based on mission outcomes. A part of the measures of effectiveness of the *crowd management in sport event mission* are: the supporters satisfaction, the risks detected and avoided, operational availability, and the global cost. The value of each MoE is also calculated. For example, the risk score is calculated using objective function from the max density,cross flows emplacement and number, weather conditions, etc. The mission-level effectiveness are performed to support the evaluation of the design solution.

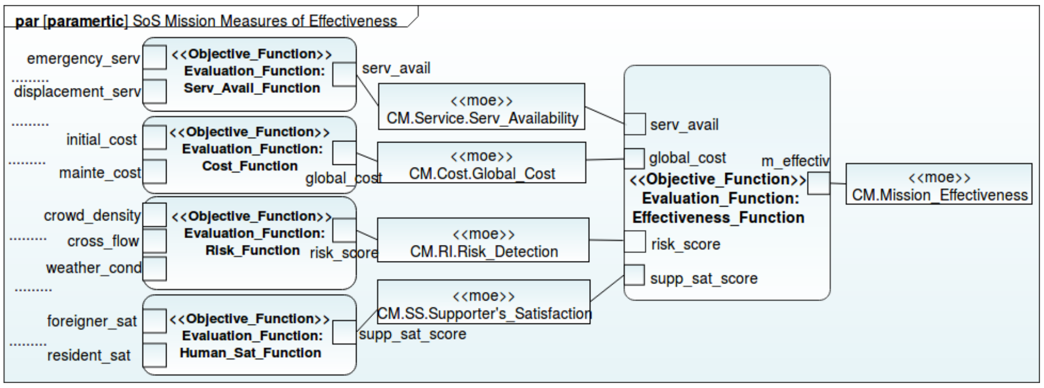

par [parametric] SoS Mission Measures of Effectiveness
emergency_serv
displacement_serv
<<Objective_Function>> Evaluation_Function: Serv_Avail_Function
serv_avail
<<moe>> CM.Service.Serv_Availability
initial_cost
mainte_cost
<<Objective_Function>> Evaluation_Function: Cost_Function
global_cost
<<moe>> CM.Cost.Global_Cost
crowd_density
cross_flow
weather_cond
<<Objective_Function>> Evaluation_Function: Risk_Function
risk_score
<<moe>> CM.RI.Risk_Detection
foreigner_sat
resident_sat
<<Objective_Function>> Evaluation_Function: Human_Sat_Function
supp_sat_score
<<moe>> CM.SS.Supporter's_Satisfaction
serv_avail
global_cost
m_effectiv
<<Objective_Function>> Evaluation_Function: Effectiveness_Function
risk_score
supp_sat_score
<<moe>> CM.Mission_Effectiveness

Figure 5.6: Mission Effectiveness Measures

5.1.4 Architecture design

Figure 5.7 and Figure 5.8 show respectively, an aggregate of roles, where each role achieves a specific *Crowd Observer in Stadium* activity or action, and the interconnection among parts that participated in the *Crowd Observer in Stadium* activity. The two diagrams are generated automatically using the ATL rules from the Crowd Observer activity diagram and the role diagram.

In the BDD (see Figure 5.7), the *In Stadium Crowd Observer* role aggregates the *Control Center* role and the seven *Observer* roles (troublemakers observer, entry points observer, etc). The seven observer roles are responsible of the *observe* mission at a particular location.

In the IBD (see Figure 5.8), the parts represent how the roles are used in the observation context and have the same role names as shown in the activity diagram. The flow ports are consistent with their definition in the activity diagram. The IBD for *In Stadium Crowd Observer* role shows the interconnection among the roles that are involved in the *In Stadium Crowd Observer* Activity Diagram. However, there is additional activity diagram that corresponds to the *Analyze Observations* activity. This

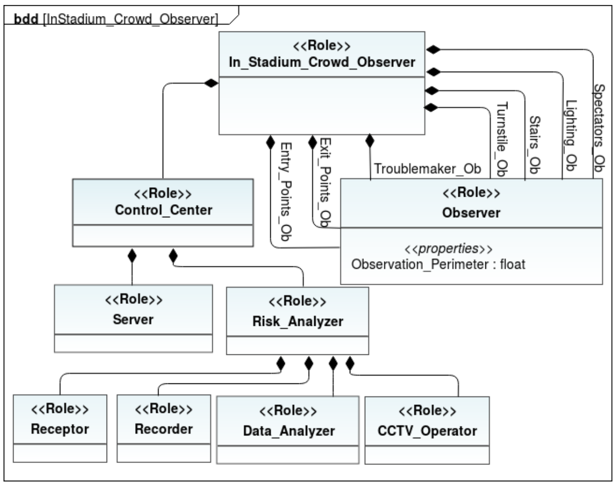


Figure 5.7: Crowd Observer in Stadium Abstract BDD

activity diagram includes different sets of interacting roles. Indeed, all the parts (roles) from all the activity diagrams are represented in the Figure 5.8. Likewise, the hierarchy of all roles is represented in the Figure 5.7. The *Control Center* aggregates the *Server* and *Risk Analyzer* roles. The *Risk Analyzer* role aggregates the *Receptor, Recorder, Data Analyzer* and the *CCTV Operator*.

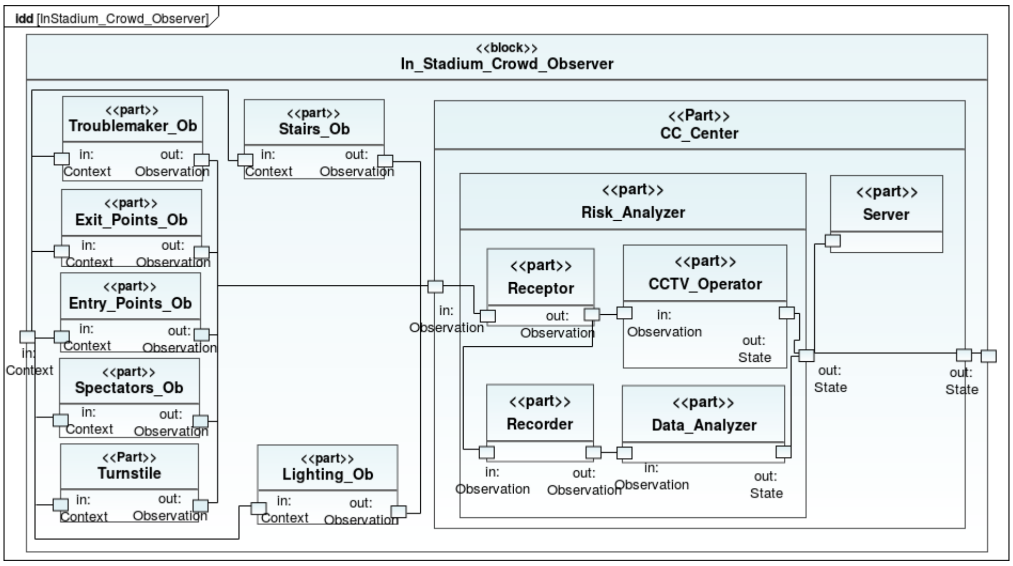


Figure 5.8: Crowd Observer in Stadium Abstract IBD

The abstract architecture is used to obtain a possible concrete architecture by replacing the abstract items with concrete ones. Each solution is characterized by a set of attributes that have a value distribution. The attributes for a given solution are then evaluated using an objective function, and the results for each alternative are compared to select the preferred solution. Figure 5.9 shows two variants of the *Observer* role serving as solution to perform the observer role in the *observe exit points* mission. The operational availability, cost, and security effectiveness are the measure of effectiveness (MoE) for the observe mission. The overall effectiveness is calculated for each alternative using a weighted equation of their MoE values.

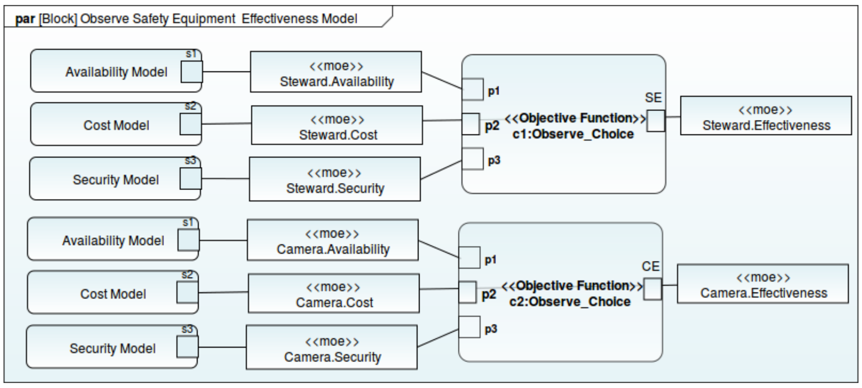

par [Block] Observe Safety Equipment Effectiveness Model
Availability Model
s1
Cost Model
s2
Security Model
s3
<<moe>> Steward.Availability
<<moe>> Steward.Cost
<<moe>> Steward.Security
p1
p2
p3
<<Objective Function>> c1:Observe_Choice
SE
<<moe>> Steward.Effectiveness
Availability Model
s1
Cost Model
s2
Security Model
s3
<<moe>> Camera.Availability
<<moe>> Camera.Cost
<<moe>> Camera.Security
p1
p2
p3
<<Objective Function>> c2:Observe_Choice
CE
<<moe>> Camera.Effectiveness

Figure 5.9: Effectiveness Evaluation Results between the two Observers Variants

According to the effectiveness of each alternative. The architect selects, evaluates and chooses the preferred concrete constituent systems. Like OOSEM, we used the concept of "physical node that represents an aggregation of physical components at a particular location". Figure 5.10, represents an example of concrete roles allocated to the abstract roles. The *Troublemakers Observer* role was assigned to *mobile steward* human concrete role, who are engaged to penetrate the crowd and observe troublemakers. The concrete architecture constrains the solution space with preselected concrete systems that are available and are able to be assembled in the SoS. When a role is allocated to software, the later must also be allocated to a corresponding hardware role to execute it. Likewise, when a role is allocated to human, the later must also be allocated to a corresponding hardware role to allow him communicate. As already mentioned and as shown in the Figure 5.10, the steward can communicate using using the Radio headsets.

The concrete architecture evaluation is performed using the overall mission effectiveness (see Figure 5.6). Simulation remains the best way to analyze the impact of an

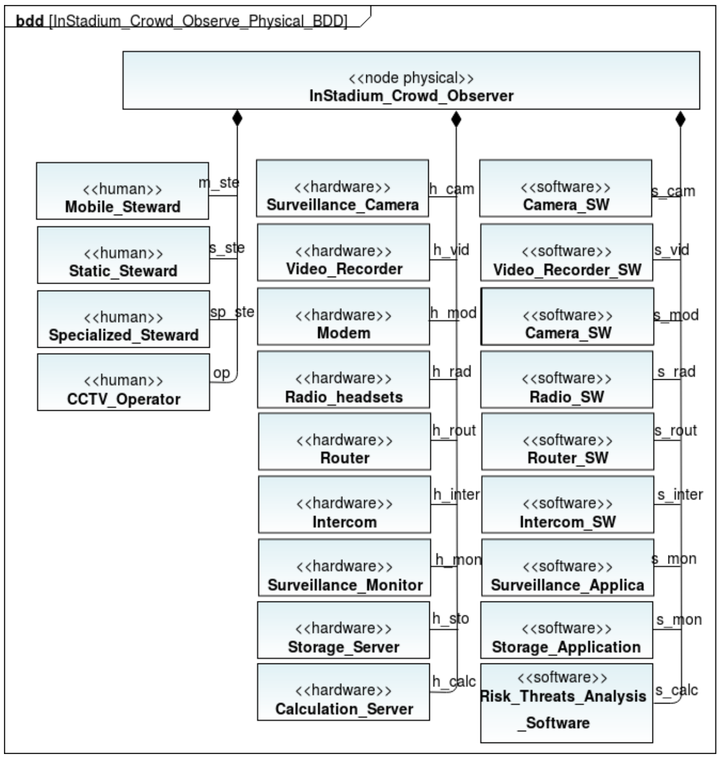

bdd [InStadium_Crowd_Observe_Physical_BDD]
<<node physical>> InStadium_Crowd_Observer
<<human>> Mobile_Steward
<<human>> Static_Steward
<<human>> Specialized_Steward
<<human>> CCTV_Operator
m_ste
s_ste
sp_ste
op
<<hardware>> Surveillance_Camera
<<hardware>> Video_Recorder
<<hardware>> Modem
<<hardware>> Radio_headsets
<<hardware>> Router
<<hardware>> Intercom
<<hardware>> Surveillance_Monitor
<<hardware>> Storage_Server
<<hardware>> Calculation_Server
h_cam
h_vid
h_mod
h_rad
h_rout
h_inter
h_mon
h_sto
h_calc
<<software>> Camera_SW
<<software>> Video_Recorder_SW
<<software>> Camera_SW
<<software>> Radio_SW
<<software>> Router_SW
<<software>> Intercom_SW
<<software>> Surveillance_Applica
<<software>> Storage_Application
<<software>> Risk_Threats_Analysis_Software
s_cam
s_vid
s_mod
s_rad
s_rout
s_inter
s_mon
s_mon
s_calc

Figure 5.10: Physical Block Definition Diagram of Crowd Observer in Stadium Node

architecture solution on mission mission effectiveness.

5.2 Crowd management simulation

Simulation represents a promising path for the analysis of systems of systems (36). While modeling allows SoS engineers to control the overall complexity of SoS, to reveal and document its structure and behavior, and to communicate these to stakeholders (44), simulation can be used to demonstrate these models from various point of view. Simulation in SoSE can be used to engineer the integration of a system into SoS (35), to avoid the undesirable emergent behavior (146, 147), to evaluate the ability of an architecture configuration to accomplish the specified SoS mission (148), or to identify the key factors that influence overall SoS performance (149).

In case of crowd management, simulation is interesting (1) to ensure that the SoS is able to give a fast response and suggest the best way to evacuate crowd safely and efficiently when emergency (such as fires and earthquakes) occurs, (2) to assist designers in evaluating risk and maximizing safe design of architectural space, (3) to detect key factors that can influence the SoS performance. Furthermore, according to (133), the complexity of crowd behaviors make several issues with crowd simulation largely open to date. The authors of (133) cites as examples 1) “the modeling of the intelligent behavior of rescue workers or the authority”, as well as 2) “modeling and simulation of panic crowds” and 3) “modeling and simulating confrontational crowd behaviors”.

In our case, we did the simulation to evaluate the SysML models that represent the concrete architecture and to analyze the overall behavior of the crowd management SoS in different event venues. The aim is to ensure that using our process, the SoS is able to respond to the defined requirements.

5.2.1 Simulation approach for SoS

According to (133, 150), SoSE’s simulation techniques can be classified into: Events-based modeling (EBM) also named discrete modeling (DEVS, Colored Petri Nets, State chart approach, etc), and agent-based modeling (ABM). While EBM represent a collection of events that impact the SOI, ABM is known for it’s use to simulate models that represent a number of entities (agents), that share a common environment usually called “word”. To represent these individual agents and their interactions, modeling these entities require a focus on detailing their characteristics, interaction rules, and behavior. Crowd management includes multiple systems to work on one goal which is to manage the crowd. Furthermore, they are mostly led by humans in real world and ABM is the best method for modeling interactions, and behavior simulation (151). In our case, each agent in our ABS represent a constituent system as a law-enforcement, security steward, or a fan etc. Our simulation approach is based on the models obtained in the analysis and design engineering activities as shown in Figure 5.11. The mission structural and behavioral model with the concrete architecture contain the concrete constituent systems, and their behavior. Thus, every constituent system becomes an agent in the environment, and its behavior becomes the agent behavior. In the mission functional model, the mission requirement are defined. In fact, the mission functional model is used to define the SoS simulation parameters. From this data, the simulation is performed and measures that indicate the actual SoS performances are collected. They are compared to those defined in the MoEs and are used to detect deficiencies in the architecture, note the lesson learned, implement updates, and take different decisions.

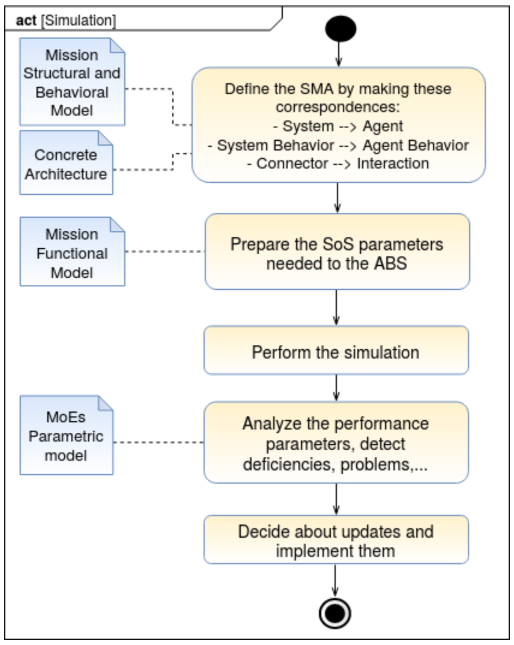

act [Simulation]
Mission Structural and Behavioral Model
Concrete Architecture
Define the SMA by making these correspondences:
- System --> Agent
- System Behavior --> Agent Behavior
- Connector --> Interaction
Mission Functional Model
Prepare the SoS parameters needed to the ABS
Perform the simulation
MoEs Parametric model
Analyze the performance parameters, detect deficiencies, problems,...
Decide about updates and implement them

Figure 5.11: Simulation Activity

5.2.2 Simulation tool

We have carried out our simulation, as part of a Master's project that we supervised, using the AnyLogic simulation tool [1], anylogic is a multi-method simulation tool that supports several simulation paradigms (discrete event, agent based, system dynamics) developed by the Anylogic company. Regarding crowd management, Anylogic is rich with simulation libraries (e.g. pedestrian library, road traffic library, etc). Anylogic is a java based platform, supporting the object oriented modeling, and the multi-threaded environment. Figure 5.12 below show a screen shot of anylogic IDE startup page.

To be able to perform the simulation of a sport event, we needed a sport arena. The sport arena used is not an official arena, but an AutoCAD [2] student's architectural design [3]. Figure 5.13 depicts the general view of the arena. We notice that there are two parking lots that can hold up to 243 cars, with two gates, the first one for car entry

[1]Simulation tool developed by the AnyLogic company. Web site: https://www.anylogic.com Accessed 14/02/2021

[2]Computer-aided design (CAD) software that architects, engineers, and construction professionals rely on to create precise 2D and 3D drawings. Web site: https://www.autodesk.eu/ Accessed 07/08/2021

[3]CAD design website. Web site: https://www.bibliocad.com/en/library/project-gym_91553/ Accessed 07/08/2021

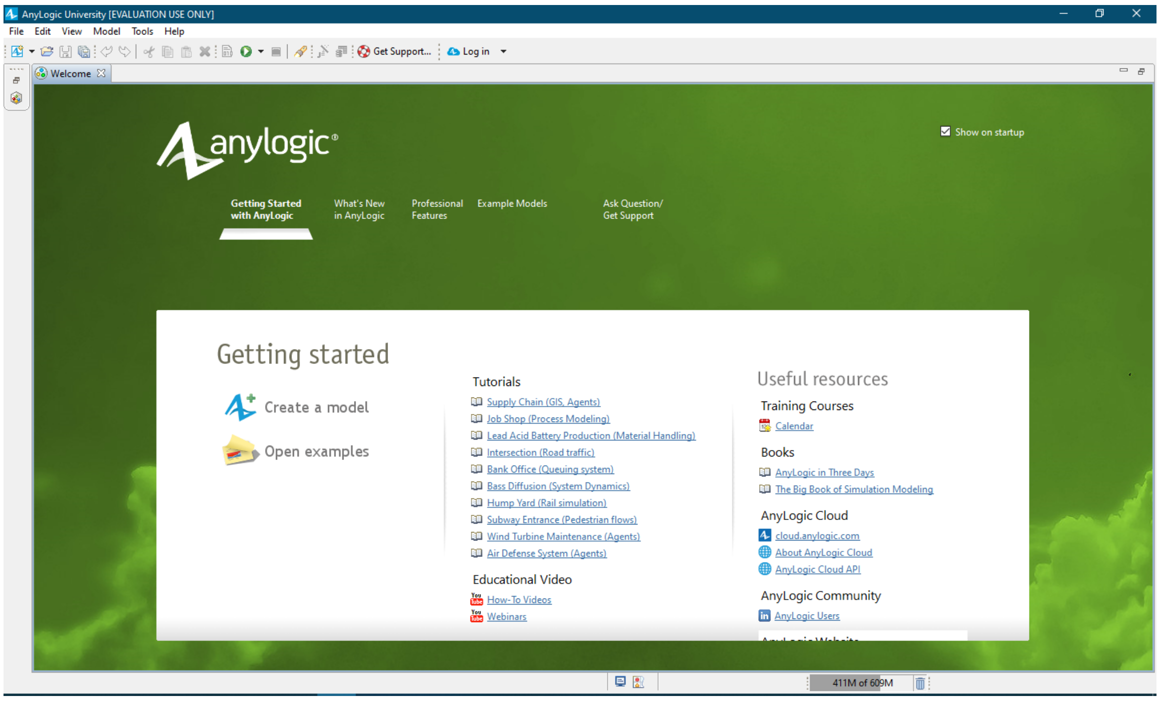

AnyLogic University [EVALUATION USE ONLY]
File Edit View Model Tools Help
Get Support... Log in
Welcome
anylogic
Show on startup
Getting Started with AnyLogic
What's New in AnyLogic
Professional Features
Example Models
Ask Question/ Get Support
Getting started
Create a model
Open examples
Tutorials
Supply Chain (GIS, Agents)
Job Shop (Process Modeling)
Lead Acid Battery Production (Material Handling)
Intersection (Road traffic)
Bank Office (Queuing system)
Bass Diffusion (System Dynamics)
Hump Yard (Rail simulation)
Subway Entrance (Pedestrian flows)
Wind Turbine Maintenance (Agents)
Air Defense System (Agents)
Educational Video
How-To Videos
Webinars
Useful resources
Training Courses
Calendar
Books
AnyLogic in Three Days
The Big Book of Simulation Modeling
AnyLogic Cloud
cloud.anylogic.com
About AnyLogic Cloud
AnyLogic Cloud API
AnyLogic Community
AnyLogic Users
411M of 609M

Figure 5.12: AnyLogic IDE

and the another one for car exit. The gates are shown in a red arrows. There are two more gates at the left of the image, the first one is the main gate for the pedestrian entrance and exit, and the other one is for special agent entrance like the teams players bus or emergency services.

Inside the arena, there are several venues that can be used by the pedestrian, namely the fast-food, bathroom, coffee. Furthermore, there is a seating area that contains 5248 seats, with multiple stairs to navigate in the area.

5.2.3 Simulation and results of crowd observation mission

Crowd observation is an important task in crowd management, as discussed in section 5.1.1. It allows to avoid several problems and risks that revolve around the crowd. That's why we chose to simulate the ***Crowd Observation*** mission. ***Observe crowd*** means that the SoSs has to ensure a safe pedestrian ***entrance to the arena***, ***inside the arena***, and a safe ***exit from the arena*** as shown in Figure 5.14. In fact, the simulation is based on three main scenarios: ***Observe crowd entrance***, ***observe crowd behavior after the entrance***, and ***observe crowd behavior when pedestrians exit the arena***. Each

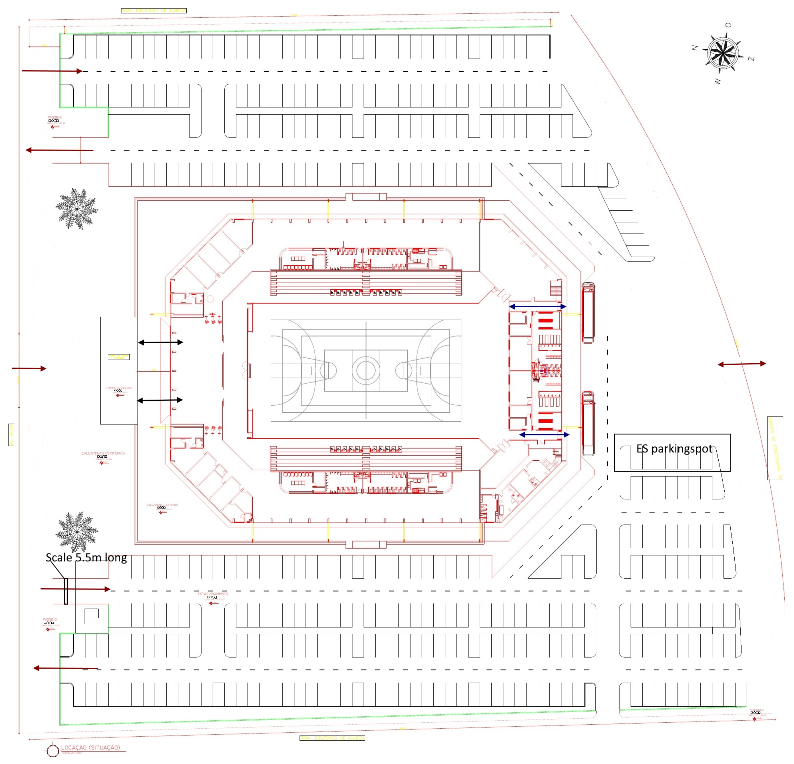
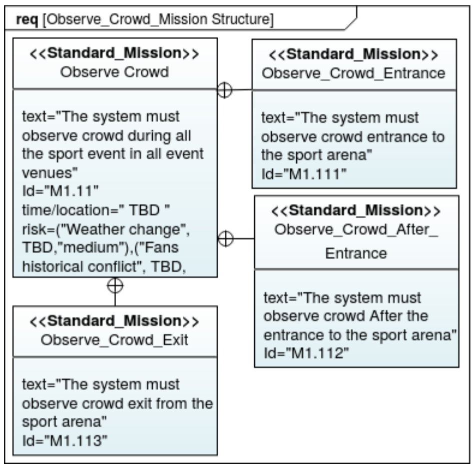


Figure 5.13: General View of the Arena

of these scenarios is presented in the following.


req [Observe_Crowd_Mission Structure]

<<Standard_Mission>>
Observe Crowd

text="The system must observe crowd during all the sport event in all event venues"
Id="M1.11"
time/location=" TBD "
risk=("Weather change", TBD,"medium"),("Fans historical conflict", TBD,

<<Standard_Mission>>
Observe_Crowd_Entrance

text="The system must observe crowd entrance to the sport arena"
Id="M1.111"

<<Standard_Mission>>
Observe_Crowd_After_Entrance

text="The system must observe crowd After the entrance to the sport arena"
Id="M1.112"

<<Standard_Mission>>
Observe_Crowd_Exit

text="The system must observe crowd exit from the sport arena"
Id="M1.113"

Figure 5.14: Observe Crowd Requirement Diagram

5.2.3.1 Crowd entrance observation mission

The first scenario in our simulation is the crowd entrance. The crowd have one main gate for pedestrian entrance, and another gate for pedestrians coming from the parking lots. Each car contains between one and five fans (random variable). The main variable to measure here is the queue time, queue time is an important factor in the crowd satisfaction criteria. The average queue time is obtained by calculating the average queue time for each fan from each venue of the arena.

In this mission, pedestrians have to safely join their seats, as fast as possible, while keeping the queue time as low as we can in the different venues. To perform a realistic simulation, several data were needed: the spawn rate of pedestrians, the movement speed, the probability for a pedestrian to enter each venue, queue line numbers. We relied on the documents cited in the section 5.1 to fix them. Some of them can be varied during simulation like the spawn rate of pedestrians, while others could not change like the pedestrian movement speed that cannot exceed 1.4m/s. The architectural drawing that we used propose 4 entry service points for each pedestrian gate, and for the arena venues we can fit 22 service points for each fast-food venue, 11 service points for each coffee venue, and 30 service points for each bathroom side. As a first result, we obtained an average queue time of 20 minutes, which is a long queue time. As a long queue time is seen today as a deficiency and a security flaw, we make changes in the SoSs models and architecture (since it is possible when using the wave model) by making possible to each fan to see on a screen or in mobile application the most suitable service point to take, the one where there are less people at the moment. Therefore, we were able to obtain an average queue time around 12 minutes as shown in Figure 5.15 and Figure 5.16 .

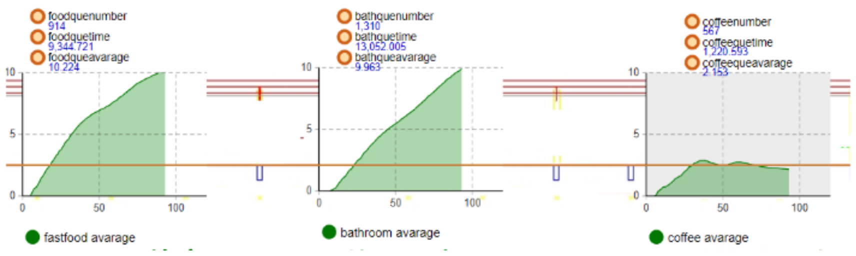


Figure 5.15: Venues's Final Average Queue Time

We can notice in Figure 5.15 that the average wait time inside the venues is around the 10 minute as marked, 10 minutes for fast-food service, 11 minutes for the bathroom, and 2 minutes for the coffee.

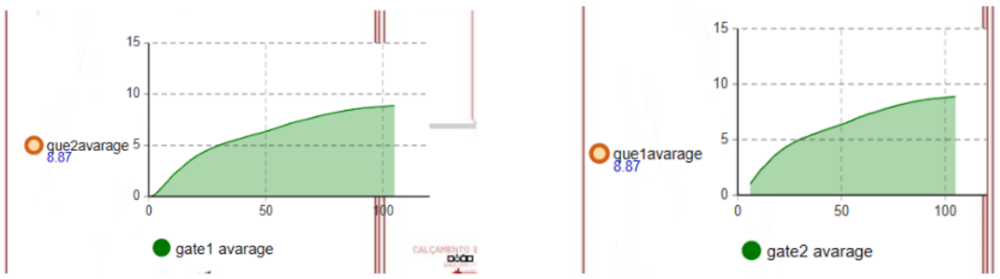
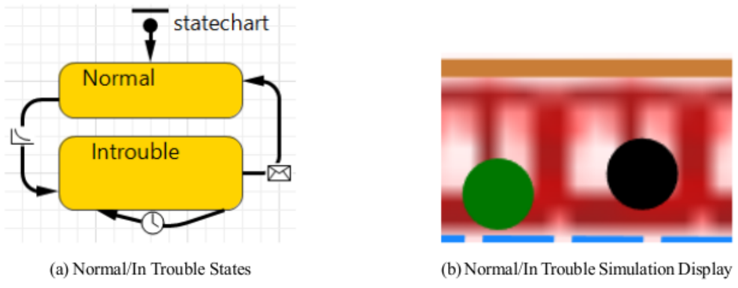


Figure 5.16: Gates Final Average Queue Time

5.2.3.2 Crowd behavior observation mission after entrance

I this scenario, we relied on the approach proposed by (152) to simulate the crowd behavior. In this case, we mean by a behavior a fan normal behavior change, so he starts seeking a mobile steward attention to him. The two states that represent the fan behavior are shown in Figure 5.17, where the state change occurs during the normal behavior of the fan following the state-chart diagram.



(a) Normal/In Trouble States (b) Normal/In Trouble Simulation Display

Figure 5.17: Fan Behavior

The Fan’s behavior state model can be described as follows:

- **Normal state**, the normal state is the default state, that the fan takes initially. The state is transformed, then into the In–trouble state with a specific rate (10% of the crowd for example).

- **In–trouble state**, in the in–trouble state, the fan that changed his state turns into a black circle as shown in figure 5.17. He starts then seeking the attention of the police officers or the mobile stewards. If the required agent is free, he can move to the troubled fan and intervene by turning the targeted fan's state to the normal.

In this scenario we are going to focus on the performance of a mobile steward in clearing up the troubles of the fan. The data we are going to capture, is the average time the fan is going to stay in in-trouble state. The simulation variable we are starting with is 30 mobile stewards spread over the seating area. The simulation variable we are starting with is 30 (1 steward/200 fans) mobile stewards spread over the seating area to cover as much as possible the space. The fan's state transformation rate depends on the nature of the crowd (calm, aggressive, etc.), we started with 20% in the course of 1 hour. Moreover, we considered the transformation time required to turn the fan back to its normal state as a random variable, which can take as a value of 30 to 40 seconds. Figure 5.18 shows an example of the simulation run, of the mobile stewards at work.

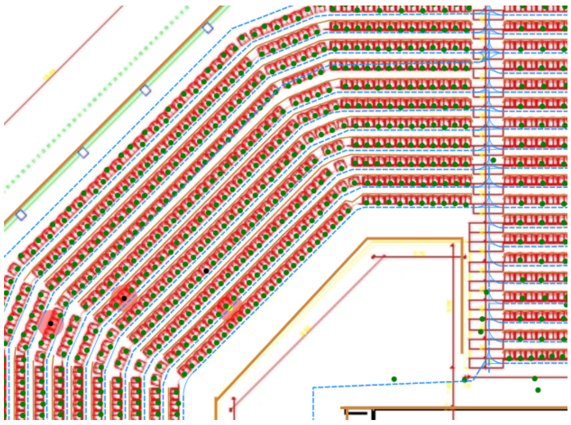


Figure 5.18: Mobile Security Steward Working, at the Seating Area

The first result obtained, as depicted in Figure 5.19, shows that the in–trouble state's average time exceeds 9 minutes/fan, and the number keep increasing as the simulation progress, with no sign of stabilization.

It is clear that by increasing the number of stewards, we can improve the result, except that the goal here is not to add more stewards, but to optimize the architecture as much as possible. We have added new constituent systems such as mobile application, that collects data from surveillance cameras, and directs the stewards movements using Dijkstra's algorithm, one of the best short path finding algorithm. Since a steward

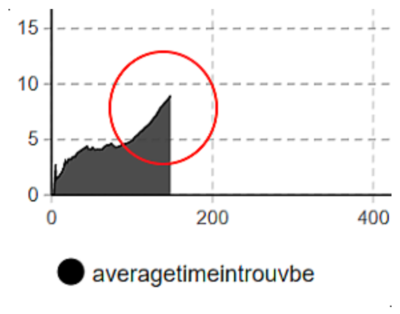


Figure 5.19: Initial Average Time In Trouble

can be accessed by multiple requests at the same time, we also improved the architecture by using in the same mobile application, a simple semaphore, that allows to control requests to the mobile stewards. In fact, when a fan is in–trouble state, the nearest free mobile steward is solicited. Thus, the new configuration is able to keep the average time of in–trouble state at 5.4 minutes/fan, the simulation ended with 1722 changed states. This result is acceptable if the effectiveness of the target mission is 5 minutes/fan. Moreover, if we continue the simulation there is no indication that the average time increases. The red circle in Figure 5.20 shows the start of a stabilization after the full amount of fans are already spawned.

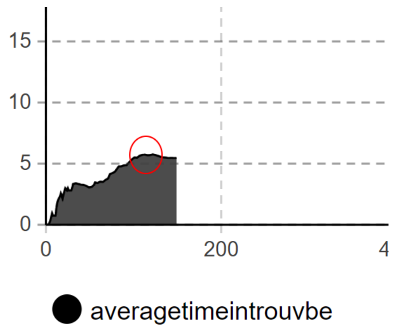


Figure 5.20: Final Average Time In Trouble

5.2.3.3 Crowd exit

Getting the crowd out of the arena safely, is one of the most important scenarios in crowd management, that if done incorrectly, disastrous consequences can happen. The main indicators that allow to judge about the crowd exit mission effectiveness is the approximate time required by the fans to exit the arena as well as their density. In this scenario, we were interested by the detection of critical areas in the arena, which are the spaces that obstruct the fans and force them to stop moving. These areas are detected by checking fans density, expressed as fans per square meter (f/m2). Figure 5.21 shows an example of a critical area.

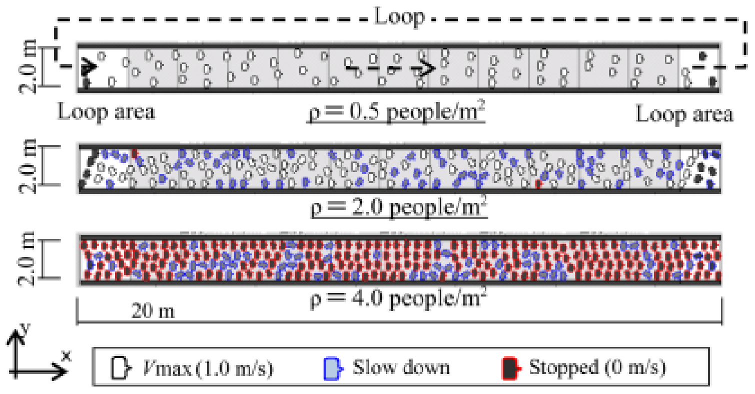


Figure 5.21: Critical Area Example (32)

In the exit scenario, each fan is going to head to the nearest exit gate to him. The time of evacuation is calculated at the moment when all fans exit the four gates. The critical areas are represented in red color filter within the simulation. Figure 5.22 illustrates the filter used to display the color of each fan density interval.

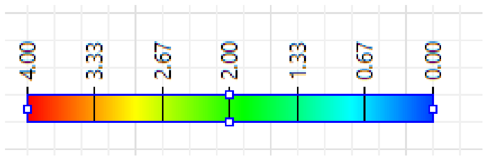


Figure 5.22: Colors Signification of the Filter Used to Display Results

After 4 minutes of running the simulation, we noticed that almost stairs have a density problem as shown in the Figure 5.23. In addition, there are two stairs (red arrows pointing at them), with the biggest density problem. That is because all the fans in the red rectangle at the left of the image, are using these stairs to exit the stadium.

When the cameras detect this kind of situations, the corrective step is initiated. It consists of managing saturated areas in order to anticipate the inherent risks. Spe-

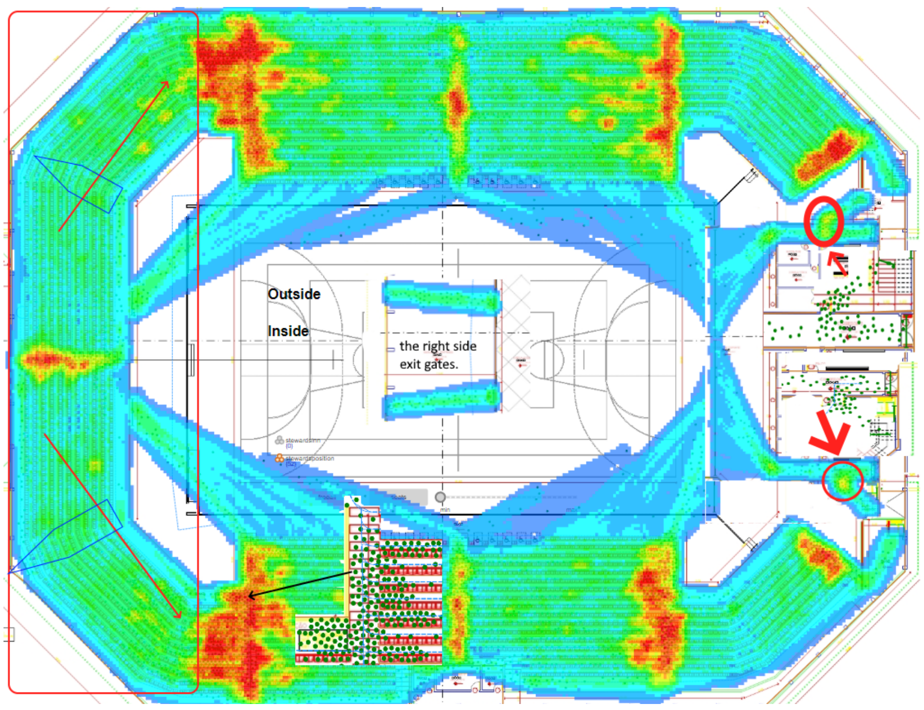


Figure 5.23: Default Arena Evacuation Simulation Results

cialized security agents take care of controlling access and managing flows. After 2 minutes of reaction of the agents, we can see in Figure 5.24 that the density is reduced. The simulation ended after 20 minutes of execution, which is acceptable timing for normal exit (not an emergency exit).

5.2.4 Discussion

In the design of crowd management systems in large-scale sports venues, the emphasis is often on safety evacuation in these venues by considering architectural (architecture of buildings) and fire-service regulations. The design of the places, depending on safety standards, the size of the corridors, stands and stairs, helps to make collective travel more fluid. Once this design has been completed, any sport event requires its organizers to take into account the potential risks and to put in place an adequate fast response for the safety of the crowd. Likewise, crowd well-being (153) and comfort, often overlooked, are essential for a successful event. Thus, during major sport event, hundred of staff members and contributors, often belonging to different organizations, are involved. Their coordination is an important factor that allows to anticipate and

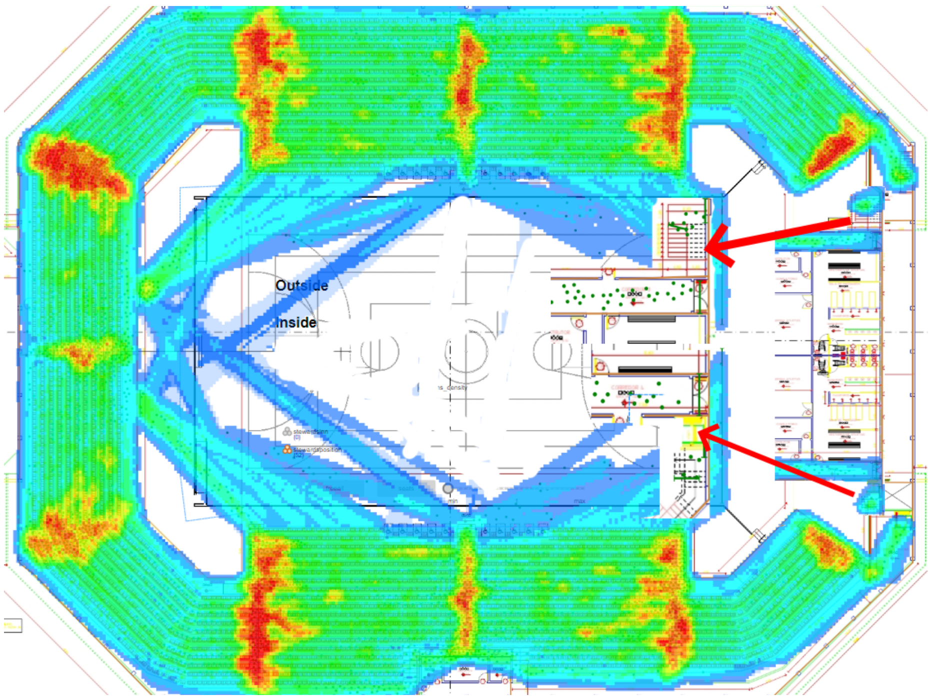


Figure 5.24: Arena Evacuation Simulation Results After Modification of the Architecture

contain these movements, that's why we proposed through this case study, the design of crowd management system, following an SoSs process, that considers the crowd management from an SoSs-wide perspective, analyzing not only the event itself, but also the potential influencing systems and elements in the SoSs environment, while maintaining the focus on mission success.

The use of our approach to design a crowd management system brings several positive points. First of all, at the level of planning and preparation. A good knowledge of the mission, its requirements, and MoEs makes possible the detection of the problems that could potentially arise, and what contingencies need to be implemented to deal with those problems.

Additionally, the definition of roles in a crowd management system allows to have a good knowledge of who are going to have a role at some point during the event, whether it is human, system, organism or another. Listing them makes perfect sense for an event, because we have constituent systems that are not used to working together usually. Thus, wide range of roles (police, mobile stewards, security stewards, event organisers, etc) used to manage the event from multiple perspectives, should aim to work together in order to enhance solidarity, and to maximize the mission effectiveness.

Mission definition in our approach allows to define the behavior and function of each role, to meet the objectives of the mission. Since the observation, vigilance and evaluation of warning signs are to be taken into consideration in such a system, different scenarios should be defined in order to manage strategies to deal with potential risks which may arise as well as the coordination of the different roles.

Being part of our approach, the simulation is useful and helpful to assess the detailed operation of any decision before its implementation under realistic conditions. Adjustments to mission strategy can be carried out depending on the results of the simulation.

Since we rely on an iterative process, constant improvements are to be made to the crowd management SoSs. Technological solutions and logistical solutions that improve the effectiveness of the mission should be adopted. Thus, several CSs are added to crowd management SoSs constantly like route planners, systems allowing supporters to have what they have ordered online delivered directly to them, and different guidance systems capable of providing information for fans on the exit gate, etc. That's why we highlight on the importance of the adoption of an SoSs wide view for crowd management systems.

CONCLUSION

In recent decades, there has been a growing interest in the development and analysis of SoS in military, industry and academic domains. While the military domain focuses on designing highly interoperable technical command and control systems to operate independently, industrial and academic interest in the investigation of SoSs architectures. Additionally, the academic perspective provides a specific and more extensive focus at the methodological aspects of SoSs development.

Both of the characteristics of an SoS (autonomy and independence of constituents, geographical distribution, and the evolutionary development), and authority relationships between the CSs and the SoS influence on how to engineer SoSs. The the US DoD outlined key differences between SE and acknowledged SoSE in four aspects (10, 33): management and oversight, operational focus, development, and engineering and design considerations. As this thesis deals with development and analysis of SoSs, these differences were considered during the research and were reformulated as challenges for any SoSE approach. First, in term of management and oversight, every process should distinguish two levels of stakeholders: CS level and SoS level stakeholders. Their priorities are different and interests may be conflicting. Second, in the operational focus aspect, SoSs are designed and developed to achieve set of operational objectives that have to be achieved using independent CSs, whose objectives may not necessarily align with the SoS objectives. Then, continuous SoS life cycle characterizes SoSs development. It is intertwined with multiple system life cycles across asynchronous acquisition and development efforts involving legacy systems, systems under development, and new developments. Finally, SoSE implies the difficulty to define SoS boundary, which is often ambiguous. It depends on the capabilities needed so that the SoS can meet its objectives. Moreover, the needed capabilities determine the CSs expected to be involved in the SoS, and the end-to-end behavior of the ensemble of systems determines the individual functionalities undertaken by the CSs to support the SoSs objectives.

In this thesis, we addressed SoSs modeling and analysis challenges already cited. We proposed a process called **Mission-Oriented Process for System of Systems Engineering (MOP-SoSE)** to overcome these challenges. We used **the mission paradigm** to balance the design and the end-to-end process. A mission has a goal, which is achieved through a sequence of operational activities. Mission Engineering (ME) determines those operational activities and allocates them to CSs for execution. We followed the guidelines of Model Driven Engineering (MDE) to develop the MOP-SoSE process using SysML (System Modeling Language) to describe the target models. We used the Emergency Response SoS as illustrative example to exhibit the process. As well, we detail and explain the different steps, and activities using the crowd management SoS case study. Finally, we perform a Multi-Agents Simulation (MAS) of the architecture to evaluate and improve the SoS mission effectiveness.

The contributions of this thesis may be summarized as follow:

Use of mission paradigm: The use of mission paradigm brings an advantage to the work. By following a mission-oriented process, the SoS engineers can develop effective architectures that focus more on the necessary capabilities of the SoS. Indeed, the SoS then achieves mission performance goals by integrating appropriate CSs into the SoS. To use mission paradigm correctly we conducted a literature review on key concepts of both SoSE and ME, and proposed then a mission conceptual model that encompasses and relates the concepts of the two domains. This conceptual model defined the basic vocabulary used in the proposed process.

Proposition of the MOP-SoSE process: This is the main contribution of this thesis, which brings many strong points:

- **Use of MBSE and SysML:** the adoption of model-based systems engineering approach brings five benefits according to the INCOSE (86): improved communications, increased ability to manage system complexity, improved product quality, enhanced knowledge capture and reuse of information, and improved ability to teach and learn SE fundamentals. Our approach has been developed to support the realization of the intended benefits of MBSE in SoS domain. SysML is a standardized system architecture modeling language. Its use results on a set of "unambiguous, structured and executable representation of the SoS architecture that can be exploited and simulated" (154).

- **Use of the wave life cycle:** the wave life cycle is suited for SoSs development because it is an evolutionary, iterative, and incremental life cycle. It is a way to outline the necessary steps for engineering a SoS. It explicitly depicts the analysis and resulting feedback throughout the entire process. We adapted the wave model to show the interaction with multiple system life cycles (CSs life cycles).
- **Strengthen the link between the SoS analysis stage and the architecture stage:** by proposing model transformation to generate the abstract architecture from the mission model. The system architect can deploy concrete constituent systems in the abstract architecture. Several concrete architecture could be obtained by replacing the roles with concrete CSs. The transformation goal is to avoid as much as possible information loss between the two stages.
- **Distinguish between SoS level stakeholders and CSs ones:** we proposed that the SoS engineers describe new CS artifacts using the requirement diagram. They are negotiated with the CS engineers to find a good compromise between CSs goals and the SoS goal, since they can be conflicting. It is to highlight that since our process is applicable for acknowledged SoS, there is active cooperation between SoS and CSs engineers.
- **Deal with the ambiguous boundary of an SoS:** by using the concept of role, abstract entities that encapsulate the ideal behavior that will fulfill an action. The concept of role is used to deal with the uncertainty of the availability of SoS constituent systems.
- **Definition of an utilization process:** to provide guidance on the use of our approach. An advantage of this process is to guide any user having basic knowledge on IT, to use the process and the proposed Domain Specific Language (DSL) to model and analyze the architecture of the SoS. The process describes step by step, the needed actors and their tasks for each engineering activity.

The proposition of MAS steps: which allows execution of the SoS architecture due to the similarities between SoS and MAS concepts such as decentralization and self organization. Such simulation enables to assess gaps in mission performance and to improve it.

Despite all the contributions cited above, taking a step back from the work done allowed us pointing some limitations on the proposed process for SoS development: It is applicable only for acknowledged SoSs, it does not support other type of SoSs development.

Perspectives

Although the work presented in this thesis covers the needs to build and manage SoSs, there is more than enough scope for future improvement on the topic.
The first future work, that may be part of our process is ***the proposition of a framework for the evaluation of SoSs architecture***, and that may be used also to compare different SoSs architectures. By framework we mean a conceptual model that identifies the different evaluation metrics for mission in SoSs context as well as the steps and rules allowing the development of those metrics. In our process, for the evaluation of the SoSs architecture, we relied on the work of (31), and used two metrics which are MoPs and MoEs. The MoPs allow to choose between two CSs that perform the same role while MoEs allow to decide about the effectiveness of the mission. In the SE literature, other important measures exist as KPPs (Key Performance Parameter) and TPMs (Technical Performance Measures). According to INCOSE (155, 156), the use of those metrics allows to have insight into the likelihood of achieving the operational objectives or capabilities, to assess the progress of the technical solution, and to evaluate the technical risk as the solution evolves. Thus, it will be interesting to show how to use them in SoSs context.

A second perspective concerns the simulation. We used basic scenarios of crowd management and a restricted set of constituent systems in our work. As our process relies on the wave model, which is iterative model, the SoS is never in its final form. Thus, the simulation has to be improved continually and should ***consider end–to–end mission real life scenarios*** (157). The connectivity to surrounding parking lots, metro stations, and bus stops are significant features of the crowd simulation in a major sporting event. Indeed, when crowd gather for an event at the same time, city's road and transportation network might quickly become overburdened. Thus, it is important at SoS level to identify those critical areas, and to find the adequate solutions to overcome them with the adequate SE teams.

Another perspective concerns the ***use of artificial intelligence in the MoP-SoSE***. Indeed, the artificial intelligence (AI) symbolized by the machine learning (ML) technology, has gained prominence in the last two decades in the SE process. Advanced AI algorithms as the deep learning (DL), and reinforcement learning (RL) provide a promising way in the prediction of critical properties if joined to modeling and simulation. They exhibit high efficiency in building automatically predictive models, in improving model performance, and personalization compared to other approaches (158). From SE perspective, several authors tried to show how AI is applicable to the development and simulation of systems as well as for the verification and validation of systems (159, 160). Indeed, it will be interesting to apply AI in the different activities of the SoSE development process, to manage mission threads, to make adequate strategical decisions, to solve problems, and to find new insight to the SoS emergent behavior that a human being may overlook.

The last, but not the least, perspective we can suggest consists in extending the current work so that ***the cyber security aspect will be taken into account at the heart of the SoSs design***. In our approach, we used the concept of role to represent any entity that may enter the SoS boundaries at a given moment. In fact, the functionalities of each role are specified and understood. Except that at runtime, whenever a CS enters the SoS boundaries, it represents a potential risk, whether intentional or not. Therefore, several cyber attack scenarios can be envisaged. We cite as an example in crowd management SoS for major sport event, the fraud on stadium tickets, fake sports betting sites, fake sites supposed to display the results of a competition, sabotage of the stadium's electrical infrastructure, denial of service attack on arbitration assistance systems, etc. The consequences of such scenarios could be significant. So, it becomes necessary to integrate the cyber security aspect as first class entity into the SoSs mission specification and design stages (161) to identify and manage vulnerabilities at an early stage, and thus provide a secure by design SoS.

Glossary

SoSE	System of Systems Engineering
SoS	System of Systems
SE	System Engineering
SDI	Strategic Defense Initiative
SoSECE	System of Systems Engineering Center of Excellence
U.S.	United States
CISA	Center for Integrated Systems in Aerospace
NCSOSE	National Centers for System of Systems Engineering
IEEE SoSE	IEEE Conference on System of Systems Engineering
IEEE SYSCON	IEEE Systems Engineering Conference
INCOSE	International Council on Systems Engineering
IJSSE	International journal of System of Systems Engineering
ISJ	IEEE Systems Journal
CS	Constituent System
ICT	Information and Communication Technology
SOI	System Of Interest

OUSD AT&L	Office of the Under Secretary of Defense for Acquisition, Technology and Logistics
DANSE	Designing for Adaptability and EvolutioN in Systems of systems Engineering
SoS SE	System of Systems System Engineering
DSL	Domain-Specific Languages
CONOPS	CONcept Of OPerationS
SysML	System Modeling Language
DoD	Department of Defense
DoDAF	Department of Defense Architecture Framework
MoDAF	Ministry of Defense Architecture Framework
NAF	NATO Architecture Framework
UPDM	Unified Profile for DoDAF and MoDAF
COMPASS	Comprehensive Modelling for Advanced Systems of Systems
CML	Compass Modelling Language
UTP	Unifying Theories of Programming
AF	Architectural Framework
AMADEOS	Architecture for Multi-criticality Agile Dependable Evolutionary Open System of Systems
ME	Mission Engineering
MIM	Mission Integration Management
NDAA	National Defense Authorization Act
MEI	Mission Engineering and Integration

NASA	National Administration of Aeronautics and Space
MMF	Missions and Means Framework
MLSoS	Mission-Level Systems of Systems
I&I	Integration and Interoperability
MoPs	Measures Of Performance
OSEs	Organizational System Elements
MT	Mission Thread
ACC	Autonomy, Collaboration and Cooperation
OODA	Observe, Orient, Decide, Act
DAG	Directed Acyclical Graph
MoP-SoSE	Mission Oriented Process for System of Systems Engineering
ADE	application domain expert
ERSoSs	Emergency Response System of Systems
MoEs	Measurements of Effectiveness
RE	Requirement
ATL	ATLAS Transformation Language
BDD	Block Definition Diagram
IBD	Internal Block Diagram
ECore	Eclipse Core
EBM	Events-Based Modeling
ABM	Agent-Based Modeling
CAD	Computer-Aided Design

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
RL	Reinforcement Learning

References

[1] International Council on Systems Engineering. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Fourth edition edition, 06 2015. ix, 10, 11, 12, 20, 25

[2] Bruce Beihoff, Christopher Oster, Sanford Friedenthal, Christiaan Paredis, Duncan Kemp, Heinz Stoewer, David Nichols, and Jon Wade. *A World in Motion âĂŞ Systems Engineering Vision 2025*. 01 2014. ix, 11, 13

[3] **ISO/IEC/IEEE International Standard - Systems and software engineering – Guidelines for the utilization of ISO/IEC/IEEE 15288 in the context of system of systems (S0S)**. *ISO/IEC/IEEE 21840:2019(E)*, pages 1–68, 2019. ix, 12, 14, 53

[4] Alex Gorod, Brian E.White, Vernon Ireland, S. Jimmy Gandhi, and Brian Sauser. *Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering*. Complex and Enterprise Systems Engineering. CRC Press; 1 edition (July 1, 2014). ix, 1, 13, 15, 101

[5] A. Gorod, B. Sauser, and J. Boardman. **System-of-Systems Engineering Management: A Review of Modern History and a Path Forward**. *IEEE Systems Journal*, **2**(4):484–499, 2008. ix, xiii, 13, 15, 28

[6] Jo Ann Lane. **What is a System of Systems and Why Should I Care?**, 2013. ix, 1, 2, 21, 22, 23

[7] Anjel Tzanev. **Modeling and Simulation of Systems of Systems – a Survey**. *Cybernetics and Information Technologies*, **13**(2):3 – 36, 2013. ix, 25, 26

[8] Charles B. Keating and Polinpapilinho F. Katina. **Systems of systems engineering: prospects and challenges for the emerging field**. *International Journal of System of Systems Engineering (IJSSE)*, **2**:234–256, 06 2011. ix, 26, 27

[9] Judith Dahmann. **Systems of Systems Engineering Life Cycle**. Educational Notes EN-SCI-276-04, NATO, S&T, January 2015. ix, 3, 32, 34, 47

[10] DEPARTMENT OF DEFENSE. *Systems Engineering Guide for Systems of Systems*, August 2008. ix, xiii, 1, 2, 20, 21, 22, 25, 27, 28, 30, 32, 33, 58, 76, 80, 123

[11] J. DAHMANN, G. REBOVICH, J. LANE, R. LOWRY, AND K. BALDWIN. **An implementers' view of systems engineering for systems of systems**. In *2011 IEEE International Systems Conference*, pages 212–217, Montreal, Canada, April 2011. ix, x, xiii, 1, 33, 34, 36, 38, 74, 75

[12] ERIC HONOUR. **Designing for Adaptability and evolution in System of systems Engineering**. In *Presented at National Defense Industry Association Systems Engineering Conference (NDIA SE)*, October 2013. ix, 34, 35

[13] IMAD SANDUKA. *A modelling framework for systems-of-systems with real-time and reliability requirements*. PhD thesis, UniversitÃd't Siegen, 2015. ix, 40

[14] SANFORD FRIEDENTHAL, ALAN MOORE, AND RICK STEINER. *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2014. ix, 2, 42, 53, 90

[15] HOFFMANN HANS-PETER. **White paper: SysML-based systems engineering using a model-driven development approach**. Technical report, October 2008. ix, 2, 43

[16] AURELIJUS MORKEVICIUS, AISTE ALEKSANDRAVICIENE, DONATAS MAZEIKA, LINA BISIKIRSKIENE, AND ZILVINAS STROLIA. **MBSE Grid: A Simplified SysML-Based Approach for Modeling Complex Systems**. *INCOSE International Symposium*, **27**(1):136–150, 2017. ix, 2, 44, 45

[17] JOHN MO AND BECKETT RONALD. *Engineering and Operations of System of Systems*. CRC Press, Boca Raton, 2019. ix, 46, 47

[18] COMPASS CONSORSIUM. **The Compass Project**. `http://www.compass-research.eu/`, 2014. ix, 49, 82

[19] STEFAN HALLERSTEDE, FINN OVERGAARD HANSEN, CLAUS BALLEGARD NIELSEN, AND KLAUS KRISTENSEN. **Guidelines for Architectural Modelling of SoS**. Research Report Delivrable D21.5a, September 2014. ix, 49, 50

[20] ARUN BABU, SORIN IACOB, PAOLO LOLLINI, AND MARCO MORI. *AMADEOS Framework and Supporting Tools*, pages 128–164. Springer International Publishing, Cham, 2016. ix, 3, 50, 51, 53

[21] Y. J. LU, L. L. CHANG, K. W. YANG, Q. S. ZHAO, AND Y. W. CHEN. **Study on System of Systems Capability Modeling Framework Based on Complex Relationship Analyzing**. In *2010 IEEE International Systems Conference*, pages 23–28, April 2010. ix, 52

[22] Andres Sousa-Poza. **Mission Engineering**. *International Journal of System of Systems Engineering*, **6**:161, 01 2015. ix, x, 3, 54, 58, 60, 61, 62, 73

[23] Robert Gold. **Mission Engineering**. In *In Proceedings of the 19th Annual National Defense Industrial Association (NDIA) Systems Engineering Conference*, October 2017. ix, 61

[24] Jack H. Sheehan, Paul H. Deitz, Britt E. Bray, Bruce A. Harris, and Alexander B.H. Wong. **The Military Missions and Means Framework**. Technical Report TR-756, U.S. Army Materiel Systems Analysis Activity, october 2004. x, 63, 64, 81

[25] Paul H. Deitz, Britt E. Bray, and James R. Michaelis. **The missions and means framework as an ontology**. In Michael A. Kolodny and Tien Pham, editors, *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR VII*, **9831**, pages 45 – 56. International Society for Optics and Photonics, SPIE, 2016. x, 64, 65

[26] R. Deiotte and R. K. Garrett. **A Novel Approach to Mission-Level Engineering of Complex Systems of Systems: Addressing Integration and Interoperability Shortfalls by Interrogating the Interstitials**. Technical Report 13-MDA-7269, Missile Defense Agency, December 2013. x, 59, 63, 64, 65, 66, 71, 72, 81

[27] Jr. James D. Moreland. **Mission Engineering Integration and Interoperability (I&I)**. Naval Surface Warfare Center, Dahlgren Division, Leading Edge, January 2015. x, 67

[28] Charles S. Wasson. *System Analysis, Design, and Development: Concepts, Principles, and Practices (Wiley Series in Systems Engineering and Management)*. Wiley-Interscience, USA, 2005. x, 59, 68, 69

[29] Eduardo Silva, Thaís Batista, and Flávio Oquendo. **A mission-oriented approach for designing system-of-systems**. In *10th System of Systems Engineering Conference*, SoSE 2015, pages 346–351, May 2015. x, 70

[30] E. Silva, E. Cavalcante, and T. Batista. **Refining Missions to Architectures in Software-Intensive Systems-of-Systems**. In *2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS)*, pages 2–8, Buenos Aires, Argentina, May 2017. x, 70, 71

[31] D. Webster, N. Looker, D. Russell, Lu Liu, and J. Xu. **An ontology for evaluation of network enabled capability.** 2008. x, 82, 87, 95, 96, 126

[32] Yoshikazu Minegishi and Naohiro Takeichi. **Design guidelines for crowd evacuation in a stadium for controlling evacuee accumulation and sequencing**. *Japan Architectural Review*, **1**(4):471–485, 2018. xi, 119

[33] JUDITH DAHMANN. **Systems of Systems Characterization and Types**. Educational Notes EN-SCI-276-01, NATO, S&T, January 2015. xiii, 1, 27, 28, 123

[34] J. DAHMANN, G. REBOVICH, J. A. LANE, AND R. LOWRY. **System engineering artifacts for SoS**. In *2010 IEEE International Systems Conference*, pages 13–17, San Diego, CA, USA, April 5-8 2010. xiii, 35, 36, 38, 80

[35] M.JAMSHIDI. *Systems of Systems Engineering: Principles and Applications*. CRC Press, 2008. 1, 17, 110

[36] CLAUS BALLEGAARD NIELSEN, GORM PETER LARSEN, JOHN FITZGERALD, JIM WOODCOCK, AND JAN PELESKA. **Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions**. *ACM Computing Survey*, **48**(2):1–18, sep 2015. 1, 2, 23, 24, 25, 82, 110

[37] MARK W. MAIER. **Architecting Principles for Systems-of-Systems**. *INCOSE International Symposium*, **6**(1):565–573, 1996. 1, 9, 11, 12, 17, 18

[38] J. BOARDMAN AND B. SAUSER. **System of Systems – the meaning of of**. In *2006 IEEE/SMC International Conference on System of Systems Engineering*, pages 6 pp.–, 2006. 1, 18, 19

[39] R. ABBOTT. **Open at the top; open at the bottom; and continually (but slowly) evolving**. In *2006 IEEE/SMC International Conference on System of Systems Engineering*, pages 6 pp.–, 2006. 1, 19, 20, 53

[40] MAIER MARK W. **Architecting principles for systems-of-systems**. *Systems Engineering*, **1**(4):267–284, 1998. 1, 21, 23

[41] PAUL N. LOWE AND MICHELLE W. CHEN. **System of Systems Complexity: Modeling and Simulation Issues**. In *Proceedings of the 2008 Summer Computer Simulation Conference*, SCSC '08, pages 36:1–36:10, Vista, CA, 2008. Society for Modeling &#38; Simulation International. 1, 91

[42] R. COLE. **The changing role of requirements and architecture in systems engineering**. In *2006 IEEE/SMC International Conference on System of Systems Engineering*, pages 6–10. IEEE, April 2006. 1

[43] J. S. DAHMANN AND K. J. BALDWIN. **Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering**. In *2008 2nd Annual IEEE Systems Conference*, pages 1–7. IEEE, 2008. 1, 21, 22, 23

[44] JIM WOODCOCK, PETER GORM LARSEN, JUAN BICARREGUI, AND JOHN FITZGERALD. **Formal Methods: Practice and Experience**. *ACM Computing Survey*, **41**(4):1–36, october 2009. 2, 110

[45] Bruce P. Douglass. **White paper: The Harmony Process**. Technical report, March 2005. 2, 43

[46] U.S. Department of Defense. **DoDAF Architecture Framework Version 2.02**. `https://dodcio.defense.gov/library/dod-architecture-framework/`, August 2010. 2, 45

[47] UK Ministry of Defence. **MOD Architecture Framework (MODAF)**. `https://www.gov.uk/guidance/mod-architecture-framework`, 2004. 2, 46

[48] NATO. **NATO Architecture Framework, Version 4.0**. Technical report, January 2018. 2

[49] Margherita Forcolin, Pietro-Felice Petrucco, Roberto Previato, Richard Lloyd Stevens, Richard Payne, Claire Ingram, and Zoe Andrew. **Accident Response Use Case Engineering Analysis Report Using Current Methods & Tools**. Technical Report D41.1, March 2014. 3, 48, 49, 82

[50] Gregg Vesonder and Dinesh Verma. **RT-171: Mission Engineering Competencies Technical Report**. Technical Report SERC-2018-TR-106, April 2018. 3, 54, 59, 63, 73

[51] A. S. Hernandez, W. D. Hatch, A. G. Pollman, and S. C. Upton. **Computer Experimentation and Scenario Methodologies to Support Integration and Operations Phases of Mission Engineering and Analysis**. In *2018 Winter Simulation Conference (WSC)*, pages 3765–3776. IEEE, 2018. 4, 59, 71

[52] **ISO/IEC/IEEE International Standard – Systems and software engineering – System of systems (SoS) considerations in life cycle stages of a system**. *ISO/IEC/IEEE 21839:2019(E)*, pages 1–40, 2019. 4, 20, 32, 71

[53] Dennis M. Buede. *The Engineering Design of Systems: Models and Methods*. Wiley Publishing, 2nd edition, 2009. 9

[54] **ISO/IEC/IEEE International Standard - Systems and software engineering – System life cycle processes**. *ISO/IEC/IEEE 15288 First edition 2015-05-15*, pages 1–118, May 2015. 10

[55] Albert Albers, Constantin Mandel, Steven Yan, and Matthias Behrendt. **System of Systems Approach for the Description and Characterization of Validation Environments**. In *2018 15th International Design Conference*, pages 2799–2810, 01 2018. 11

[56] Stefan N. Grösser. *Complexity Management and System Dynamics Thinking*, pages 69–92. Springer International Publishing, Cham, 2017. 12

[57] Ciza Thomas, Rendhir R. Prasad, and Minu Mathew. **Introduction to Complex Systems, Sustainability and Innovation**. In Ciza Thomas, editor, *Complex Systems, Sustainability and Innovation*, chapter 1. IntechOpen, Rijeka, 2016. 12

[58] Kenneth E. Boulding. **General Systems Theory – The Skeleton of Science**. *Management Science*, **2**(3):197–208, 1956. 13

[59] Russell L. Ackoff. **Towards a System of Systems Concepts**. *Management Science*, **17**(11):661–671, 1971. 14

[60] Michael Jackson and P. Keys. **Towards a System of Systems Methodologies**. *Journal of the Operational Research Society*, **35**:473–486, 06 1984. 14

[61] Francois Jacob. *The logic of living systems : a history of heredity / [by] Francois Jacob ; translated [from the French] by Betty E. Spillmann*. Allen Lane London, 1974. 14

[62] United States. *Restructuring of the Strategic Defense Initiative (SDI) Program*. S. hrg. ;100-1010. U.S. G.P.O., Washington, 1989. iii, 86 p. 14

[63] Howard Eisner, John Marciniak, and Ray McMillan. **Computer-aided system of systems (C2) engineering**. In *IEEE International Conference on Systems, Man, and Cybernetics*, 1991. 16

[64] Aaron J. Shenhar. **2.5.1 A New Systems Engineering Taxonomy**. *INCOSE International Symposium*, **5**(1):723–732, 1995. 16

[65] A. J. Shenhar and Z. Bonen. **The new taxonomy of systems: toward an adaptive systems engineering framework**. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, **27**(2):137–145, 1997. 16

[66] John H. Holland. *Hidden Order: How Adaptation Builds Complexity*. Addison Wesley Longman Publishing Co., Inc., USA, 1996. 17

[67] John R. Koza. **Hidden Order: How Adaptation Builds Complexity.** *Artificial Life*, **2**(3):333–335, 1995. 17

[68] Vadim Kotov. **Systems of systems as communicating structures**. Computer Systems Laboratory, HPL-97-124, 1997. 17

[69] Dominique Luzeaux and Jean-René Ruault. *Systems of Systems*. ISTE Ltd and John Wiley & Sons, Inc, 2010. 18, 80

[70] W. C. Baldwin and B. Sauser. **Modeling the characteristics of system of systems**. In *2009 IEEE International Conference on System of Systems Engineering (SoSE)*, pages 1–6, 2009. 19

[71] W. C. Baldwin, B. J. Sauser, and J. Boardman. **Revisiting "The Meaning of Of" as a Theory for Collaborative System of Systems**. *IEEE Systems Journal*, **11**(4):2215–2226, 2017. 19

[72] DEPARTMENT OF DEFENSE (DOD). *Defense Acquisition Guidebook.* Washington, D.C.: U.S. Dept.of Defense, Pentagon, February 2010. 20

[73] DEPARTMENT OF DEFENSE (DOD). *Defense Acquisition Guidebook.* Washington, D.C.: U.S. Dept.of Defense, Pentagon, October 2004. 20

[74] **ISO/IEC/IEEE International Standard - Systems and software engineering – Taxonomy of systems of systems**. *ISO/IEC/IEEE 21841:2019(E)*, pages 1–20, 2019. 20, 21

[75] CHAIRMAN OF THE JOINT CHIEFS OF STAFF (CJCS). *CJCS Manual 3170.01C, "Operation of the Joint Capabilities Integration and Development System"*, May 2007. 21

[76] ANDREW P. SAGE AND CHRISTOPHER D. CUPPAN. **On the Systems Engineering and Management of Systems of Systems and Federations of Systems**. *Inf. Knowl. Syst. Manag.*, **2**(4):325–345, december 2001. 25

[77] BAR-YAM YANEER, ANN ALLISON MARY, BATDORF RON, CHEN HAO, GENERAZIO HOA, SINGH HARCHARANJIT, AND TUCKER STEVE. **The Characteristics and Emerging Behaviors of System of Systems**, 2004. 25

[78] CHARLES KEATING, RALPH ROGERS, RESIT UNAL, DAVID DRYER, ANDRES SOUSA-POZA, ROBERT SAFFORD, WILLIAM PETERSON, AND GHAITH RABADI. **System of Systems Engineering**. *Engineering Management Journal*, **15**(3):36–45, 2003. 25

[79] J. O. CLARK. **System of Systems Engineering and Family of Systems Engineering from a standards, V-Model, and Dual-V Model perspective**. In *2009 3rd Annual IEEE Systems Conference*, pages 381–387, 2009. 32

[80] JUDITH DAHMANN, JO LANE, G. REBOVICH, AND KRISTEN BALDWIN. **A model of systems engineering in a system of systems context**. In *Conference on Systems Engineering Research*, April 2008. 33

[81] JO ANN LANE AND JUDITH S. DAHMANN. **Process Evolution to Support System of Systems Engineering**. In *Proceedings of the 2Nd International Workshop on Ultra-large-scale Software-intensive Systems*, ULSSIS '08, pages 11–14, New York, NY, USA, 2008. ACM. 33, 53

[82] DAVID. DOMBKINS. *Complex project management : seminal essays / by David H. Dombkins.* BookSurge Publishing North Charleston, S.C, 2007. 34

[83] DANSE PROJECT. **DANSE Methodology V03**. Technical report, IAI, February 2015. 34

[84] *Overview of Architecture Frameworks and Modeling Languages for Model-Based Systems Engineering*, **Volume 2: 31st Computers and Information in Engineering Conference, Parts A and B** of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 08 2011. 39, 40

[85] INCOSE. **Systems Engineering Vision 2020, version 2.03**. INCOSE Technical Operations INCOSE-TP-2004-004-02, International Council on Systems Engineering, Seattle, WA, 2007. 39

[86] SANFORD FRIEDENTHAL, REGINA GRIEGO, AND MARK SAMPSON. **INCOSE Model Based Systems Engineering (MBSE) Initiative**. In *INCOSE 2007 Symposium*, June 2007. 40, 124

[87] **IEEE Standard Glossary of Software Engineering Terminology**. *IEEE Std 610.12-1990*, pages 1–84, 1990. 40

[88] **IEEE Recommended Practice for Architectural Description for Software-Intensive Systems**. *IEEE Std 1471-2000*, pages 1–30, 2000. 40

[89] CANTOR MURRAY. **White paper: Rational Unified Process for Systems Engineering, RUP SE, Version 2.0**. Technical report, May 8 2003. 44

[90] MICHEL D. INGHAM, ROBERT D. RASMUSSEN, MATTHEW B. BENNETT, AND ALEX C. MONCADA. **Engineering complex embedded systems with State Analysis and the Mission Data System**. In *AIAA Journal of Aerospace Computing, Information and Communication*, pages 507–536, 2004. 44

[91] J. WOODCOCK, A. CAVALCANTI, J. FITZGERALD, P. LARSEN, A. MIYAZAWA, AND S. PERRY. **Features of CML: A Formal Modelling Language for Systems of Systems**. In *2012 7th International Conference on System of Systems Engineering (SoSE)*, pages 1–6, July 2012. 48

[92] C.A.R. HOARE AND J. HE. *Unifying Theories of Programming*, **14**. Prentice Hall, 1998. 49

[93] PERRY SIMON AND HOLT JON. **Definition of the COMPASS Architectural Framework Framework**. Research Report Delivrable D21.5b, December 2014. 49

[94] RUSSELL LOCK AND IAN SOMMERVILLE. **Modeling and Analysis of Socio-Technical System of Systems**. In *Proceedings of the 15th IEEE International Conference on Engineering of Complex Computer Systems*, ICECCS '10, pages 224–232. IEEE Computer Society, March 2010. 51, 54, 81

[95] WERNER DAMM AND ALBERTO SANGIOVANNI VINCENTELLI. **A Conceptual Model of System of Systems**. In *Proceedings of the Second International Workshop on the Swarm at the Edge of the Cloud*, SWEC '15, pages 19–27, New York, NY, USA, 2015. ACM. 52

[96] CHARLES B. KEATING, JOSE J. PADILLA, AND KEVIN ADAMS. **System of Systems Engineering Requirements: Challenges and Guidelines**. *Engineering Management Journal*, **20**(4):24–31, 2008. 53, 55

[97] ALEXANDER KOSSIAKOFF, SAMUEL J. SEYMOUR, DAVID A. FLANIGAN, AND STEVEN M. BIEMER. *System of Systems Engineering*, chapter 20, pages 583–596. John Wiley & Sons, Ltd, 2020. 54

[98] EILEEN BJORKMAN. **Mission Engineering for Warfighting Integration of Net-Centric Systems**. In *13th Annual Systems Engineering Conference*, October 2010. 58

[99] KATHLEEN GILES AND KRISTIN GIAMMARCO. **A mission-based architecture for swarm unmanned systems**. *Systems Engineering*, **22**(3):271–281, 2019. 59

[100] ALEJANDRO HERNANDEZ, TAHMINA KARIMOVA, AND DOUGLAS NELSON. **Mission Engineering and Analysis: Innovations in the Military Decision Making Process**. In *In Proceedings of the American Society for Engineering Management 2017 International Annual Conference*, 10 2017. 59

[101] DOUGLAS VAN BOSSUYT, PAUL BEERY, BRYAN O'HALLORAN, ALEJANDRO HERNANDEZ, AND EUGENE PAULO. **The Naval Postgraduate School's Department of Systems Engineering Approach to Mission Engineering Education through Capstone Projects**. *Systems*, **7**:38, 08 2019. 59

[102] BEERY AND PAULO. **Application of Model-Based Systems Engineering Concepts to Support Mission Engineering**. *Systems*, **7**(3):44, Sep 2019. 59

[103] U.D.O. DEFENSE. *Dictionary of Military Terms and Acronyms*. Praetorian Press LLC, 2011. 59

[104] DEPARTMENT OF DEFENSE (DOD). *Mission Engineering and Integration (MEI) Guidebook*. Washington, D.C.: U.S. Dept.of Defense, Pentagon, November 2019. 59

[105] DEPARTMENT OF DEFENSE (DOD). *Defense Acquisition Guidebook*. Washington, D.C.: U.S. Dept.of Defense, Pentagon, September 2020. 59

[106] ROBERT GOLD. **Mission Engineering**. In *In Proceedings of the 19th Annual National Defense Industrial Association (NDIA) Systems Engineering Conference*, October 2016. 61, 62

[107] THOMAS IRWIN. *Operational Mission Architecture Framework: A Blended Architecture Methodology for Enabling Operational Capability*. Thèse, Naval Pstgraduate School, Monterey, California, 2018. 61

[108] DEPARTMENT OF DEFENSE (DOD). *Defense Critical Infrastructure Program (DCIP): DoD Mission-Based Critical Asset Identification Process (CAIP)*. Washington, D.C.: U.S. Dept.of Defense, Pentagon, May 2017. 62

[109] R. GIACHETTI, S. WANGERT, AND R. ELDRED. **Interoperability Analysis Method for Mission-Oriented System of Systems Engineering**. In *2019 IEEE International Systems Conference (SysCon)*, pages 1–6. IEEE, 2019. 62

[110] William D. Miller Hoong Yan See Tao Dinesh Verma Gregg T. Vesonder Nicole A.C. Hutchison, Sergio Luna and Jon Patrick Wade. **Mission Engineering Competencies**. 63, 67

[111] John R. Boyd. **Destruction and Creation**. 1976. 66

[112] John R. Boyd. **A Discourse on Winning and Losing**. Technical report, Air University Press, 2018. 66

[113] James Surowiecki. *The Wisdom of Crowds*. Anchor, 2005. 66

[114] JM Held. **Systems of Systems: Principles, Performance, and Modelling**. *Sydney, Australia*, 2008. 66

[115] Nicole Hutchison, Hoong Yan See Tao, William Miller, Dinesh Verma, and Gregg Vesonder. **Framework for Mission Engineering Competencies**. *28th Annual INCOSE International Symposium*, **28**(1):518–531, July 2018. 67

[116] Charles S. Wasson. *System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices*. Wiley Series in Systems Engineering and Management. Wiley, 2 edition, 2015. 68

[117] Imane Cherfa, Nicolas Belloir, Salah Sadou, Régis Fleurquin, and Djamal Bennouar. **Systems of systems: From mission definition to architecture description**. *Systems Engineering*, **22**(6):437–454, 2019. 74

[118] G. Muller. **Are stakeholders in the constituent systems SoS aware? Reflecting on the current status in multiple domains**. In *2016 11th System of Systems Engineering Conference (SoSE)*, pages 1–5. IEEE, 2016. 77

[119] I. Cherfa, S. Sadou, N. Belloir, R. Fleurquin, and D. Bennouar. **Involving the Application Domain Expert in the Construction of Systems of Systems**. In *2018 13th Annual Conference on System of Systems Engineering (SoSE)*, pages 335–342, Paris, France, June 2018. 82

[120] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. **Tropos: An Agent-Oriented Software Development Methodology**. *Autonomous Agents and Multi-Agent Systems*, **8**(3):203–236, may 2004. 83

[121] Axel van Lamsweerde. **Requirements Engineering: From Craft to Discipline**. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 238–249, New York, NY, USA, 2008. ACM. 83

[122] Object Management Group. **Systems Modeling Language V1.5**. Technical Report formal/2017-05-01, Object Management Group, http://www.omg.org/spec/SysML/1.5/, 2017. 83, 84, 90

[123] Office Of The Under Secretary Of Defense For Acquisition Technology And Logistics. *Risk Management Guide for DOD Acquisition, 6th Edition (Version 1.0).* Defense Technical Information Center, WASHINGTON DC., 2006. 85

[124] International Organization for Standardization. *ISO 31000 Risk management - Guidelines*, 2 edition, February 2018. 85

[125] Michel Dos Santos Soares, Jos Vrancken, and Alexander Verbraeck. **User requirements modeling and analysis of software-intensive systems**. *Journal of Systems and Software*, **84**(2):328 – 339, 2011. 85

[126] Noel Sproles. **Coming to grips with measures of effectiveness**. *Systems Engineering*, **3**(1):50–58, 2000. 86

[127] National Research Council. *Making the Soldier Decisive on Future Battlefields*. The National Academies Press, Washington, DC, 2013. 87

[128] U.S. Air Force Doctrine. *Air Force Doctrine Publication 3-0 - Operations and Planning*. Air University, Montgomery, United States, November 2016. 87

[129] Department of the Army. **The Conduct Of Information Operations**. Technical Report ATP 3-13.1, Washington, DC, October 2018. 87

[130] Nik Looker, David Webster, Duncan Russell, and Jie Xu. **Scenario Based Evaluation**. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 148–154. IEEE, 2008. 87

[131] Esmond N. Urwin, Colin C. Venters, Duncan J. Russell, Lu Liu, Zongyang Luo, David E. Webster, Michael Henshaw, and Jie Xu. **Scenario-based design and evaluation for capability**. In *2010 5th International Conference on System of Systems Engineering*, pages 1–6. IEEE, 2010. 87

[132] Ian Sommerville. *Software Engineering*. Pearson, 10th edition, 2015. 95

[133] M.-L Xu, Hao Jiang, Xiaogang Jin, and Zhigang Deng. **Crowd Simulation and Its Applications: Recent Advances**. *Journal of Computer Science and Technology*, **29**:799–811, 09 2014. 102, 111

[134] A. M. Al-Shaery, S. S. Alshehri, N. S. Farooqi, and M. O. Khozium. **In-Depth Survey to Detect, Monitor and Manage Crowd**. *IEEE Access*, **8**:209008–209019, 2020. 102

[135] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012. 102

[136] FÉDÉRATION INTERNATIONALE DE FOOTBALL ASSOCIATION. **Stades de Football Recommandations et Exigences Techniques**. Technical report, FIFA, 2011. 102

[137] FÉDÉRATION INTERNATIONALE DE FOOTBALL ASSOCIATION. **Stadium Safety and Security Regulations**. Technical report, FIFA, 2018. 102

[138] ROSE SHEPHERD, CHRISTOPHER CLEGG, AND MARK ROBINSON. *Understanding Crowd Behaviours, Volume 1: Practical Guidance and Lessons Identified.* 04 2010. 102

[139] GREAT BRITAIN. CABINET OFFICE, LEEDS UNIVERSITY BUSINESS SCHOOL, AND EMERGENCY PLANNING COLLEGE. *Understanding Crowd Behaviours: Practical guidance and lessons identified.* Understanding Crowd Behaviours. TSO, 2010. 102

[140] GOVERNMENT OF INDIA NATIONAL DISASTER MANAGEMENT AUTHORITY. **Managing Crowd at Events and Venues of Mass Gathering**. Technical report, NDMA, 2014. 102

[141] GROUPEMENT DES INDUSTRIES DE DÉFENSE ET DE SÉCURITÉ TERRESTRES ET AÉROTERRESTRES. **Gestion des foules**. Technical report, GICAT, 2018. 102

[142] GROUPE DE TRAVAIL SUR LA CONSERVATION DES COLLECTIONS DU SOUS-COMITÉ DES BIBLIOTHÈQUES. **Guide d'Élaboration d'un Plan d'Urgence**. Technical report, BibliothÃÍque nationale du Québec, Montréal, 2010. 102

[143] LE BUREAU DE LA GESTION DES SITUATIONS D'URGENCE DE LA VILLE DU GRAND SUDBURY. **Plan d'Intervention en Cas d'Urgence**. Technical report, Ville du Grand Sudbury, 2014. 102

[144] SÉCURITÉ PUBLIQUE CANADA (SP). **Guide pour la Planification de la Gestion des Urgences**. Technical report, Government of Canada, 2018. 102

[145] LONDON EMERGENCY SERVICES LIAISON PANEL (LESLP). *Major Incident Procedure Manual.* TSO, 2012. 102

[146] X. JIAN, G. BING-FENG, Z. XIAO-KE, Y. KE-WEI, AND C. YING-WU. **Evaluation method of system-of-systems architecture using knowledge-based executable model**. In *2010 International Conference on Management Science Engineering 17th Annual Conference Proceedings*, pages 141–147, Nov 2010. 110

[147] RYMEL BENABIDALLAH, SALAH SADOU, ARMEL ESNAULT, AND MOHAMED AHMED NACER. **Simulating systems of systems using situation/reaction paradigm**. *Concurrency and Computation: Practice and Experience*, **n/a**(n/a):e4921, 2018. 110

[148] W. ROSS, M. ULIERU, AND A. GOROD. **A multi-paradigm modelling simulation approach for system of systems engineering: A case study**. In *2014 9th International Conference on System of Systems Engineering (SOSE)*, pages 183–188, June 2014. 110

[149] G. MULLER AND C. DAGLI. **Simulation for a coevolved system-of-systems meta-architecture**. In *2016 11th System of Systems Engineering Conference (SoSE)*, pages 1–6, June 2016. 110

[150] W. CLIFTON BALDWIN, BRIAN SAUSER, AND ROBERT CLOUTIER. **Simulation Approaches for System of Systems: Events-based versus Agent Based Modeling**. *Procedia Computer Science*, **44**:363 – 372, 2015. 2015 Conference on Systems Engineering Research. 111

[151] FRANZISKA KLUGL AND ANA L. C. BAZZAN. **Agent-Based Modeling and Simulation**. *AI Magazine*, **33**(3):29, Sep. 2012. 111

[152] JONATHAN OZIK, NICHOLSON COLLIER, TODD COMBS, CHARLES M. MACAL, AND MICHAEL NORTH. **Repast Simphony Statecharts**. *Journal of Artificial Societies and Social Simulation*, **18**(3):11, 2015. 116

[153] JIE LI, H. RIDDER, A. VERMEEREN, C. CONRADO, AND CLAUDIO MARTELLA. **Designing for crowd well-being: Current designs, strategies and future design suggestions**. In *5th International Congress of International Association of Societies of Design Research, IASDR 2013*, page 2278âĂŞ2289, 08 2013. 120

[154] J. DAHMANN, A. MARKINA-KHUSID, A. DOREN, T. WHEELER, M. COTTER, AND M. KELLEY. **SysML executable systems of system architecture definition: A working example**. In *2017 Annual IEEE International Systems Conference (SysCon)*, pages 1–6, Montreal, Quebec, Canada, April 2017. 125

[155] JIM OAKES, RICK BOTTA, AND A. TERRY BAHILL. **11.1.1 Technical Performance Measures**. *INCOSE International Symposium*, **16**:1466–1474, 07 2006. 126

[156] J.ROEDLER GARRY AND JONES CHERYL. **Technical Measurement Guide**. Technical Report INCOSE-TP-2003-020-01, San Diego, CA, USA, December 2005. 126

[157] RYMEL BENABIDALLAH, SALAH SADOU, AND MOHAMED AHMED-NACER. **Using System of Systems' States for Identifying Emergent Misbehaviors**. In *27th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2018, Paris, France, June 27-29, 2018*, pages 66–71. IEEE Computer Society, 2018. 126

[158] LUCY HUTCHINSON, BERNHARD STEIERT, ANTOINE SOUBRET, JONATHAN WAGG, ALEX PHIPPS, RICHARD PECK, JEAN-ERIC CHAROIN, AND BENJAMIN RIBBA. **Models and Machines: How Deep Learning Will Take Clinical Pharmacology to the Next Level**. *CPT: Pharmacometrics & Systems Pharmacology*, **8**(3):131–134, 2019. 127

[159] F.LAWLESS WILLIAM, MITTU RANJEEV, A.SOFGE DONALD, SHORTELL THOMAS, AND MCDERMOTT TOM. *Systems Engineering and Artificial Intelligence*. Springer International Publishing, 2021. 127

[160] B. Badiru Adedeji. *Artificial Intelligence and Digital Systems Engineering*. CRC Press, 2021. 127

[161] Nan Messe, Vanea Chiprianov, Nicolas Belloir, Jamal El Hachem, Régis Fleurquin, and Salah Sadou. **Asset-Oriented Threat Modeling**. In *19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2020, Guangzhou, China, December 29, 2020 - January 1, 2021*. IEEE, 2020. 127