UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA - CCET
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

HEURISTICS FOR SYSTEMS-OF-SYSTEMS DESIGN

Marcio Imamura

Orientador

Rodrigo Pereira dos Santos

RIO DE JANEIRO, RJ - BRASIL

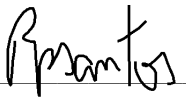
Setembro 2021

HEURISTICS FOR SYSTEMS-OF-SYSTEMS DESIGN

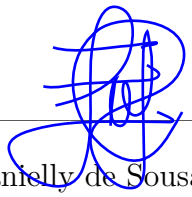
Marcio Imamura

DISSERTAÇÃO DE MESTRADO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

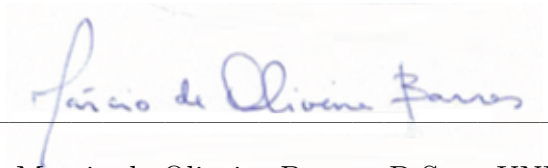
Aprovada por:



Rodrigo Pereira dos Santos, D.Sc. - UNIRIO



Everton Ranielly de Sousa Cavalcante, D.Sc. - UFRN



Marcio de Oliveira Barros, D.Sc. - UNIRIO

RIO DE JANEIRO, RJ - BRASIL

Setembro 2021

Catalogação informatizada pelo(a) autor(a)

I31 Imamura, Marcio
Heuristics for Systems-of-Systems Design / Marcio Imamura. -- Rio de Janeiro, 2021.
136 f.

Orientador: Rodrigo Pereira dos Santos.
Dissertação (Mestrado) - Universidade Federal do Estado do Rio de Janeiro, Programa de Pós-Graduação em Informática, 2021.

1. Systems-of-Systems. 2. Heuristics . 3. Systems modeling. I. Santos, Rodrigo Pereira dos, orient. II. Título.

Acknowledgements

First and foremost, I thank God for blessing my life and allowing me to get here with joy and health.

I am extremely grateful to my supervisor, Prof. Rodrigo Pereira dos Santos, who made this work possible. His diligent guidance and insightful advice took me through all stages of my research with confidence, being always present and respectful throughout this study journey.

I want to thank Francisco Ferreira for being a hard-working researcher who helped me a lot and who knew when to say "stay calm" in the most desperate moments. I would also like to thank Juliana Fernandes for their providential support.

I would like to thank all Complex Systems Engineering Laboratory colleagues for helping me and for their friendship. In special, Luiz Costa, Nadja Piedade, Luciana Chueri, and Felipe Cordeiro.

I want to give special thanks to my wife Ananda for her unfailing support and understanding over the last two years while she worked twice as hard to raise our son. I want to thank my son Antonio for his frequent interruptions asking me to play, always keeping me focused on what's essential in this world. They both had a warm smile that comforted me at difficult moments.

I would like to thank my parents, Mr. Antorio (*in memoriam*), and Mrs. Luzia (*in memoriam*). Both were raised in modest homes and understood how to pursue education for themselves, me and my three brothers, Eduardo, Ricardo, and Fernando. They provided us all we needed to be a happy family and decent persons.

Finally, I would like to thank IBGE for supporting me during the last two years.

IMAMURA, Marcio. **Heurísticas para Projeto de Sistemas-de-Sistemas**. UNIRIO, 2021. 136 páginas. Dissertação de Mestrado. Programa de Pós-Graduação em Informática, UNIRIO.

Resumo

Um sistema-de-sistemas (SoS) é um arranjo de sistemas independentes que trabalham em sinergia para cumprir missões que nenhum desses sistemas poderia realizar isoladamente. SoS podem ser observados em vários domínios, como mobilidade urbana, saúde e cidades inteligentes, para citar alguns. Uma preocupação significativa dos engenheiros de SoS se refere a independência dos sistemas constituintes que tem autonomia para parar de contribuir ou abandonar um SoS, o que dificulta garantir a qualidade do que é entregue em tempo de design. O objetivo deste estudo é investigar boas práticas e recomendações que podem ser aplicadas ao design de SoS para garantir sua adequada operação. Para tanto, esse trabalho utiliza o termo "heurísticas" para descrever tais práticas e recomendações. Foi conduzido um estudo exploratório para entender quais as preocupações em relação a um SoS operando em uma organização pública brasileira. Além disso, foi conduzido um mapeamento sistemático da literatura (MSL) para identificar quais as heurísticas que vêm sendo aplicadas em design de SoS. Um grupo focal foi realizado para organizar os resultados do MSL. Por fim, foi aplicada uma pesquisa de opinião a especialistas para avaliar quais heurísticas eram apropriadas ao design de SoS, resultando em um catálogo de heurísticas. Foi criada uma ferramenta que incorpora algumas das heurísticas do catálogo para verificar como as heurísticas podem facilitar o processo de design de SoS. Um estudo de viabilidade foi conduzido com profissionais para avaliar a facilidade de uso e a utilidade da ferramenta. Espera-se que o catálogo de heurísticas e a ferramenta possam apoiar pesquisadores e profissionais no processo de design de SoS e ajudem a identificar questões críticas durante a fase de design.

Palavras-chave: Heurísticas, Sistemas-de-Sistemas, Modelagem de sistemas

IMAMURA, Marcio. **Heuristics for Systems-of-Systems Design**. UNIRIO, 2021. 136 pages. Master's Thesis. Graduate Program in Informatics, UNIRIO.

Abstract

A system-of-systems (SoS) is an arrangement of independent systems that work in synergy to fulfill missions that any of these systems in isolation cannot accomplish. SoS could be observed in several domains such as urban mobility, healthcare, and smart cities, to mention a few. A significant concern of SoS engineers refers to the independence of the constituent systems that have the autonomy to stop contributing or abandon an SoS, making it difficult to guarantee the quality of SoS at design time. This study aims to investigate good practices and recommendations that can be applied to the design of SoS to assure its proper operation. Therefore, we herein adopted the term "heuristics" to refer to such good practices and recommendations. An exploratory study was conducted to understand the concerns regarding an SoS operating in a Brazilian public organization. We further conducted a systematic mapping study (SMS) to identify which practices have been applied in SoS design. The results were discussed in a focus group to organize the first set of heuristics. Finally, we surveyed experts to evaluate which heuristics are appropriate to SoS design, resulting in a heuristics catalog. To facilitate the understanding and implementation, the heuristics in this catalog have been organized into groups. Each heuristic was categorized according to its suitability for use based on the coordination level of the SoS. A tool was created that incorporates some of the heuristics from catalog to verify verify how the heuristics can facilitate the SoS design process. A feasibility study was conducted with practitioners to evaluate the ease of use and the usefulness of the tool. It was expected that the heuristics catalog and the tool would support researchers and professionals in the SoS design process and help identify critical issues during the design phase.

Keywords: Heuristics, Systems-of-Systems, Systems modeling

Contents

List of Figures

List of Tables

Chapter 1. Introduction

This chapter presents the context, motivation, problem, objectives and methodology of this research. We also present the structure of this dissertation.

1.1 Context

Organizations seek to boost productivity to promote competitiveness and social development. In this context, information systems (IS) are essential resources for companies and governments to accomplish their goals. An IS is an organized combination of people, hardware, software, communication networks, data, policies, and procedures. It stores, transform, and disseminates information in an organization (O'BRIEN et al., 2011), also supporting business and decision-making processes.

New technologies such as cloud computing, Internet of Things (IoT), and microservices have fostered new possibilities to develop IS by providing infrastructure that simplifies the development of new solutions with a faster development cycle, safer environments, and lower cost. Such a scenario increases the presence of software systems in the organizations and the integration of such a multitude of systems enables the emergence of new capabilities.

This scenario have led professionals to adopt solutions referred to as systems-of-systems (SoS), which is based on the collaboration among systems. SoS are characterized by the managerial and operational independence of the systems that are part of them, called constituent systems (CS). CS are often developed and maintained by different teams.

Remarkable examples of SoS are smart cities (BOSCARIOLI et al., 2017), in which public and private organizations work in coordination through the integration of systems. SoS are also present in several areas such as healthcare, public security, traffic, and other public and private services.

Nevertheless, there are several difficulties and open problems related to SoS that require advances in system' engineering (JAMSHIDI, 2008). In addition, there are several concerns on how to design and maintain SoS in a more secure, reliable, and efficient way since the CS are independent.

1.2 Motivation

The independence of the CS can be a challenge in SoS. In traditional systems, the operation of the systems' components is fully controlled by only one entity. In SoS, there is more than one decision-maker, and the CS are not necessarily known at design time (OQUENDO, 2016), which can make the architecture highly dynamic and complex. CS can experience changes in the life cycle and be modified according to their individual goals. However, there is a lack of standards to deal with the challenges imposed by the SoS specific characteristics.

1.3 Problem

The process of designing an SoS can raise questions that are difficult to respond to, such as:

- Which are the human and material resources needed?

  Integrate systems from different companies, which are geographically dispersed, and each one has individual goals, can be difficult, especially in defining human and financial resources correctly.

- Which and whose responsibilities are involved in the processes?

  A set of new responsibilities will be necessary for the SoS to work correctly. However, in an environment in which everyone has their own goals, it can be difficult to identify the responsibilities.

- Which resources are necessary for the SoS to work correctly?

  It can be challenging to define, negotiate and guarantee how resources will be delivered and consumed and how the SoS will use them.

- How to monitor the interaction among the independent systems?

> In SoS, the independence of the CS produces a dynamic behavior of the architecture. Hence, it is necessary to monitor the interactions among the CS to take appropriate actions in case of deviations.

There are several questions involving the design of SoS. Managers, developers, and users should clearly understand how the SoS will operate and which systems are interoperating. In addition, there are concerns on how to mitigate possible problems that can be foreseen at design time and define the resources necessary to develop and maintain the SoS before it starts operating.

1.4 Objective and Research Questions

This study aims to develop a catalog of heuristics by identifying rules, guidelines, good practices, and recommendations that can be applied to the design of SoS. We expect the catalog to help professionals to design SoS with more predictability and better chances of success.

We defined a research question to guide our research and two secondary questions to help us organize, evaluate, and understand the data collected in our research. The primary research question is:

> PRQ. *Which heuristics can be applied to the design of SoS?*

Heuristics are efficient cognitive processes, conscious or unconscious, which ignore part of the information (GIGERENZER; GAISSMAIER, 2011) and can be used in several areas of knowledge as a tool to provide:

- A good approximation rather than the perfect solution to a problem;
- A "well-calibrated guess" for decision making;
- Objective form of evaluation without having to know in depth the problem.

The Merriam-Webster's Dictionary [1] defines heuristics as "an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods". Herein, we are considering heuristic as **a recommendation to a designer in the**

[1] https://www.merriam-webster.com/dictionary/heuristic

form of a set of empirical rules for identifying SoS issues to be resolved at design time, when the cost of making wrong decisions is lower.

The secondary questions are:

- RQ1. For which types of SoS the heuristics can be applied?

  Rationale: One of the issues to be considered in the design of SoS is how CS are coordinated to provide the capabilities in a collaborative arrangement. The level of coordination over the CS can affect the feasibility of applying heuristics. Thus, the purpose is to identify to which types of SoS each heuristic can be applied.

- RQ2. Are there interdependent relationships between the heuristics?

  Rationale: Certain practices may affect the adoption of others. The purpose of this question is to verify if there are trade-offs on the application of heuristics.

As a secondary objective, we developed a tool that allows the professionals to model an SoS using the mKAOS notation and check if the model is correct according to the catalog of heuristics.

1.5 Research Methodology

We followed the methodology shown in Figure 1.1. The steps are described following:

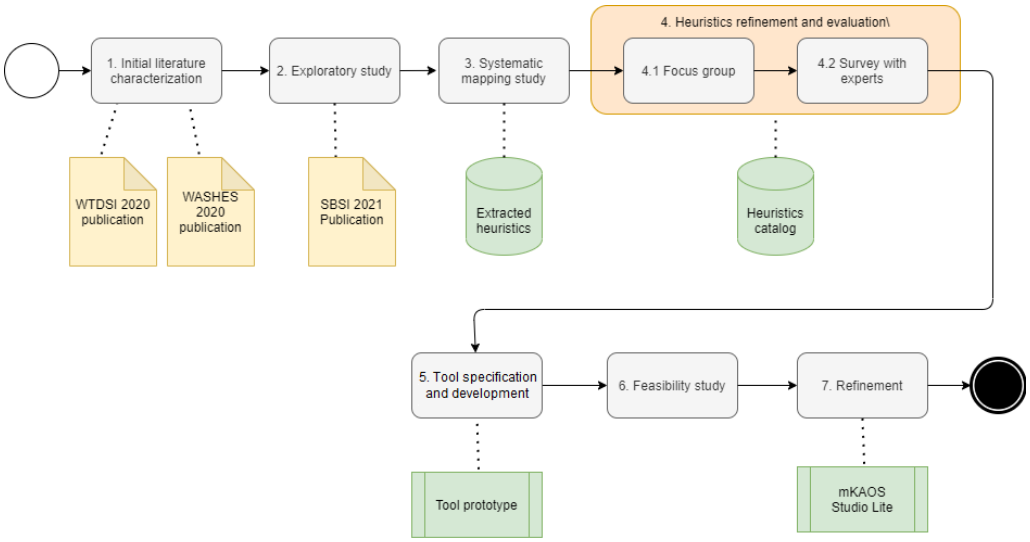


Figure 1.1: Research methodology.

1. **Literature review characterization.** The objective of this step is to understand the SoS concepts and the challenges in the research field. This phase allowed to define the scope of the research. In this phase it was published a paper in the Extended Proceedings of XIII Workshop on Theses and Dissertations in Information Systems (WTDSI'20) (IMAMURA; FERREIRA; SANTOS, 2020). It was also published a study in the Proceedings of the V Workshop on Social, Human and Economic Aspects of Software (WASHES'20) that investigates factors to be considered when implementing SoS governance (IMAMURA; COSTA, et al., 2020).

2. **Exploratory study.** In this phase, we conducted an exploratory study in a Brazilian public organization to verify the perception of professionals on issues related to a SoS in operation. Our study was published in the Proceedings of the XVII Brazilian Symposium of Information Systems (SBSI'21) (IMAMURA; FERREIRA; FERNANDES, et al., 2021).

3. **Systematic mapping study (SMS).** The study aimed to identify the challenges of designing SoS reported in the literature and the approaches to solve them. In this phase, we extracted the first set of heuristics.

4. **Heuristics refinement and evaluation.** We divided this phase into two. First, we conducted a focus group to discuss the usefulness and applicability of the heuristics identified in the SMS. Then, we refined the catalog based on the focus groups discussion and carried out a survey with experts in a second round of evaluation.

5. **Tool specification and development**. In this phase, we identified the requirements for a tool that implemented the heuristics within an mKAOS model and defined the appropriate infrastructure to support the tool.

6. **Feasibility study.** It was conducted a feasibility study with professionals from the industry to verify the ease of use and usefulness of the tool. This phase also allowed us to identify errors and make improvements in the tool.

7. **Refinement.** The first version of the tool was deployed using the feedback from the feasibility study. We named the tool mKAOS Studio Lite in reference

to the mKAOS Studio (SILVA; BATISTA, et al., 2015a).

1.6 Organization

This dissertation is organized as follows.

Chapter 2 presents the fundamental concepts of SoS, modeling, and heuristics. We also describe how heuristics have been applied in other areas.

Chapter 3 presents the exploratory study we conducted to obtain a preliminary understanding of how industry professionals perceive an SoS. We identified the problems that affect the SoS operation and how they have been overcome.

Chapter 4 details the methodology we adopted to develop and evaluate the heuristics catalog.

In Chapter 5, we explain the process of development and evaluation of the mKAOS Studio Lite.

In Chapter 6, the conclusion and contributions are presented. We also describe the research limitations and discuss the future work.

Chapter 2. Background

This chapter presents the fundamentals of SoS, the mKAOS language, and the fundamentals of heuristics. We also present a systematic mapping study regarding heuristics to SoS design.

2.1 Systems-of-Systems

In the last few years, the rise of the Internet of Things (IoT), virtual reality, machine learning, microservices, cloud computing, and other technologies has fostered the emergence of disruptive solutions, such as for Industry 4.0, smart environments, and e-health, to name a few. These solutions have in common the fact that they integrate independent systems into larger and more complex systems that provide capabilities that can only be achieved through collaboration among systems. Such arrangements are sometimes called System-of-Systems (SoS).

The literature has reported an increasing interest in SoS in the last few years (CADAVID et al., 2020). In particular, the Brazilian Information Systems community presented several challenges regarding SoS engineering that will be faced in the following years (BOSCARIOLI et al., 2017).

The term SoS is not new. In 1956, Kenneth Boulding described SoS as "a whole that is perceived as more than the sum of its parts" (BOULDING, 1956). Ackoff defined SoS as an organization containing at least two systems that have a common purpose. In this organization, systems react to behavior and communication with another system (ACKOFF, 1971). The term has also been used by François Jacob, in the 1970s, in the context of biological systems to describe "every object that is studied by biology" (JACOB, 1974). Some other definitions of SoS are presented below to show that there is no consensus on this concept:

"System-of-systems exist when there is a presence of a majority of the

> *following five characteristics: operational and managerial independence, geographic distribution, emergent behavior, and evolutionary development."* (JAMSHIDI, 2008)
>
> *"Systems of systems are large-scale concurrent and distributed systems that are comprised of complex systems."*(CARLOCK et al., 2001)

;

> *"In relation to systems to support war fighting, a system of systems is the integration of advanced command, control, computers, communications, and information systems with intelligence, surveillance, and reconnaissance systems to provide dominant battle space awareness."* (MANTHORPE, 1996)

.

> *"A system would be termed a "system-of-systems" or a "collaborative system" when: (1) its components fulfilled valid purposes in their own right and continued to operate to fulfill those purposes if disassembled from the overall system, and (2) the components systems are managed (at least in part) for their own purposes rather than the purposes of the whole."* (MAIER, 1998)

.

> *"A large widespread collection or network of systems functioning together to achieve a common purpose."* (SHENHAR, 1994)

.

The idea of SoS from the perspective of systems engineering, as a set of interconnected independent systems, began in the 1980s with the US Defense Strategy (NIELSEN et al., 2015). From that moment on, the field gained notoriety both in academia and in industry. Systems-of-Systems Engineering (SoSE) emerged as a recognized discipline for facing SoS-related challenges.

The SoS approach is verified in healthcare, power grids, aerospace, and smart cities, to name a few. However, although there are numerous researches on SoS,

there are still divergences regarding its concept (NIELSEN et al., 2015). This is due there is confusion regarding the distinction between SoS and "monolithic" systems". Asif Mahmood, for example, argues that this fact produces consequences in the differentiation between systems engineering and SoSE. The author also highlights that there are understandings that still make it more difficult to characterize this class of systems (MAHMOOD, 2016):

> *"The human body is not an SoS, although it contains multiple systems such as nervous system, respiratory system, digestive system, etc. The rationale being that detaching eye or hand from the body cannot function purposefully."*

Mahmood affirms that there is a severe concern in distinguishing SoS from monolithic systems when prominent researchers consider a dog as an SoS (BAR-YAM, 2004) and the human body as the best example of SoS (CLARK, 2009).

As a more accurate form to distinguish types of systems with their respective characteristics, the author proposes a flow of complexity represented in Figure 2.1. The boxes represent the types of systems, and the balloons describe the criteria for distinguishing such systems. The farther to the right of the image, the more complex is the system (MAHMOOD, 2016).

2.1.1 Systems-of-Systems Characteristics

Due to the difficulty in conceptualizing SoS, researchers started to define its fundamental characteristics. A widely accepted characterization comes from Mark Maier (MAIER, 1996), recognized by the International Council of Systems Engineering as relevant in the sense of defining what it is an SoS. Maier defined that the five essential characteristics of SoS are:

1. Operational independence of CS, which means that the SoS is formed by systems that are independent and useful by itself;

2. Managerial independence of CS, meaning that CS are acquired and managed separately, maintaining their lifecycle independent of the SoS;

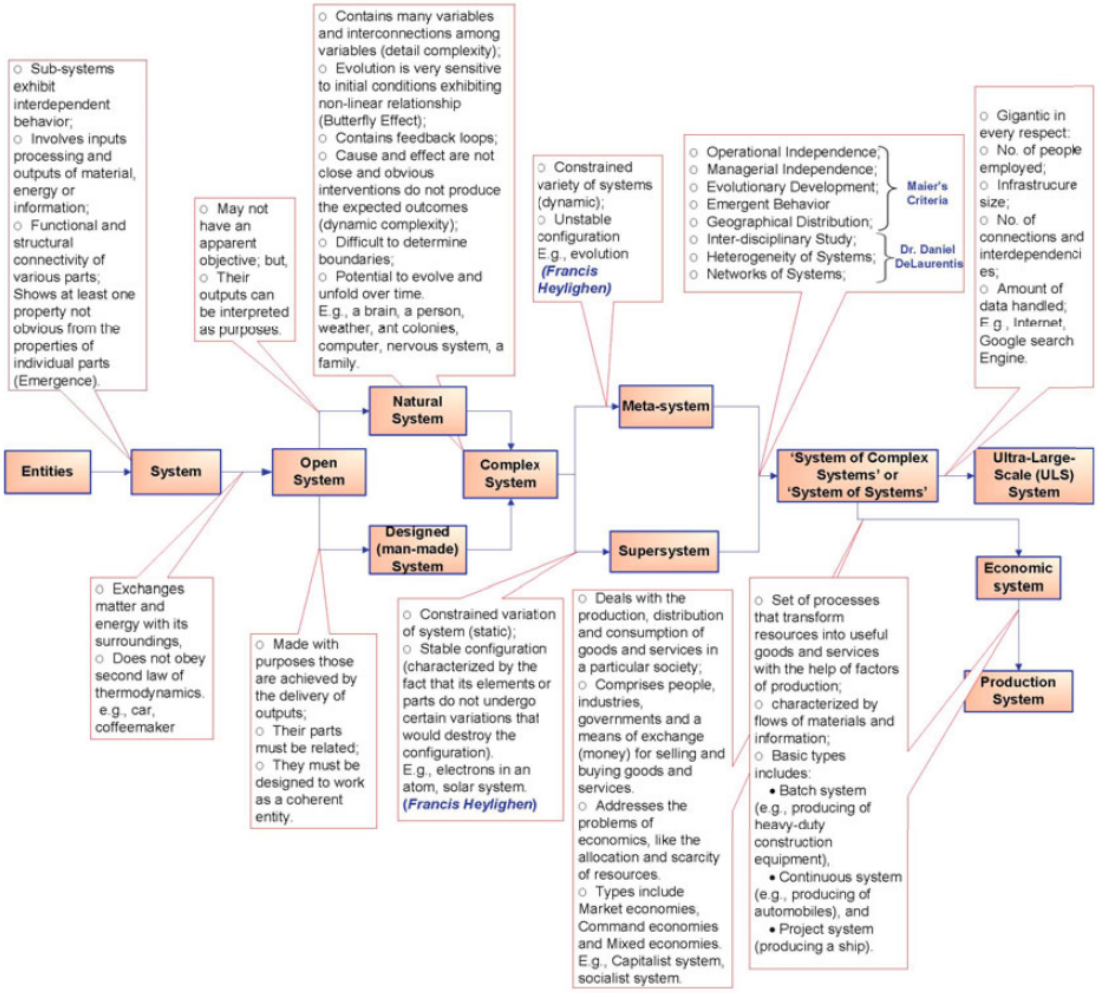


Figure 2.1: Framework to categorize systems (MAHMOOD, 2016).

3. Evolutionary development, by which SoS works and the purposes and implementations are added, removed and modified with experience;

4. Emergent behavior, where the SoS performs functions and carries out capabilities that do not reside in any of the CS. The principal purposes of the SoS are fulfilled by these emergent behaviors; and

5. Geographical distribution, indicating that CS are not operating in the same location and depend on communication links to exchange information.

Despite this, the literature suggested that other features exist and that they should be considered, as they are important for SoS Engineering. Hence, Claus Nielsen proposed eight main dimensions of SoS based on a broad survey of the literature (NIELSEN et al., 2015): **i)** autonomy of the CS; **ii)** independence; **iii)**

distribution; **iv)** evolution; **v)** dynamic reconfiguration; **vi)** emergent behavior; **vii)** interdependence; and **viii)** interoperability.

The **autonomy** is related to the measure in which the behavior of a CS is governed by its own rules and not by external ones. The CS perform their functions according to individual rules even when serving the mission of an SoS (MAIER, 1996). Autonomy is related to the ability of a CS itself to pursue a specific goal (BOARDMAN et al., 2006).

Due to the diversity of scenarios in which SoS are deployed, there may be variations in the autonomy exhibited by the CS. Therefore, modeling and analysis techniques should allow the expression of such variations and should make explicit the actions a CS can accomplish (NIELSEN et al., 2015).

Independence is the ability of a CS to operate out of its SoS. This characteristic implies that a CS offers capabilities related to its role in an SoS and capabilities specific to it by itself, i.e., they are independent of the SoS. That is, a CS exists by itself and does not depend on the SoS. Independence of both design and operation is a critical element in the definition of SoS (SHARAWI et al., 2006).

Distribution refers to the dispersion of the CS so that some connectivity mechanism must exist to allow communication and the sharing of information. This characteristic denotes a physical separation, whether it is big or not. SoS models must have the ability to describe the interconnection of CS through a communication channel. As a result, concurrency, delays, and communication failures must be considered when modeling.

The **evolution** is related to the fact that SoS change over time, either in capabilities provided, in quality of the services provided, or even in its composition. Thus, the evolutionary nature of SoS is an essential characteristic of this class of complex systems. Moreover, Carlock et al. (2001) argue that SoS does not have a permanent state, and Abbott (ABBOTT, 2006) emphasizes that SoS evolves continuously. The evolutionary nature of SoS stems from the autonomy of its CS, which have their development cycles, and the interventions that happen are manifestations of individual needs associated with the environment.

Dynamic reconfiguration is the ability of SoS to make changes in its structure and composition. Several authors consider the ability to self-change an im-

portant characteristic, especially for ensuring that an SoS can handle failures and other threats at runtime. To ensure dynamic reconfiguration ability, SoS models should have abstractions for dynamic modification of architectures and interfaces (NIELSEN et al., 2015).

Emergent behavior is the behavior that is the result of synergistic collaboration among the CS. These behaviors are emergent properties of the entire SoS and not the behavior of any of the CS (SAGE et al., 2001). However, emergent behaviors can be classified as desired or undesired. Therefore, the modeling and analysis tools should allow the verification of emergent behaviors so that a practitioner can maximize the effects of desired emergent behavior and avoid, or at least minimize, the effects of the undesired behavior.

Interdependence consists of mutual dependence between CS. If a CS depends on others CS to accomplish a task, there is an interdependence. Some authors understand that an SoS should establish trade-offs between the level of independence of the CS and the interdependence required to fulfill the SoS objective. It is essential to highlight that, although the CS are independent and autonomous, the interactions and interoperability require some degree of interdependence.

Finally, **interoperability** is the ability of an SoS to incorporate a set of heterogeneous CS that can exchange and understand information. Interoperability among the CS must be present at the syntactic and semantic levels (MADNI et al., 2014). Syntactic interoperability involves integrating and adapting communication interfaces, protocols, and standards to enable the interaction between systems. Semantic interoperability consists of the ability of two or more systems to understand the meaning of information. In SoS, CS must share and respect a common vocabulary so they can achieve interoperability semantics.

2.1.2 Types of Systems-of-Systems

SoS are categorized according to the level of authority over the CS. SoS can be virtual, collaborative, directed, or acknowledged (MAIER, 1996; DAHMANN et al., 2008). The types directed, collaborative, and virtual were defined in 1996 by Mark Maier. The type "acknowledged" was defined in 2008 by Dahmann and Baldwin.

The categories indicate the existence of a level of control, which exerts influence

on the architecture of an SoS, as it determines how adaptable and cooperative each CS will be concerning requirements, interfaces, formats of data, and technologies (NIELSEN et al., 2015).

In a **directed** SoS there is a central authority that ensures that such goals are accomplished. The operational independence of the CS is preserved, but the operation is subject to a central authority. Directed SoS typically exist in organizational environments such as the military.

A **virtual** SoS has no management authority and no common purpose. Virtual SoS are characterized by a high degree of emergence. The forms and structures that produce the capabilities of a virtual SoS are complicated to distinguish. The World Wide Web (WWW) is an example of virtual SoS, in which control is associated with published standards such as name resolution, navigation, and document structure. Websites choose to adopt or not the published patterns. The virtual SoS is controlled by forces that facilitate the adoption of standards (in the WWW, W3C[1] has this mission), which do not evolve in a controlled way but emerge according to the success it achieves among the users.

In a **collaborative** SoS, CS are not required to comply with a central authority. Instead, they voluntarily collaborate in order to achieve the goals of SoS. In a collaborative SoS, there is the idea of shared management, but with very limited or non-existent powers to enforce decisions. As an example of collaborative SoS, it is possible to cite the Internet (MAIER, 1996), in which the IETF[2] (Internet Engineering Task Force) defines standards and protocols but has no power of imposition. The Internet began as a directed SoS, controlled by the United States Advanced Research Projects Agency Network. Over time, this network evolved from centralized control to a decentralized environment, using unplanned collaboration mechanisms.

In an **acknowledged** SoS, there are agreed goals, a designated manager, and resources specific to the SoS. The CS maintain independence in ownership, objectives, financing, and development, and support. The acknowledged SoS focuses on establishing collaborative management, maintaining administrative and technical independence at the CS level. The objective is that autonomy and ownership are maintained and, at the same time, guarantee that changes can be collaborative

[1] http://www.w3.org
[2] http://www.ietf.org

and decided based on some common goals. A smart city, where different independent agencies communicate in order to improve the quality of life of citizens, is an excellent example of an acknowledged SoS (NCUBE et al., 2018).

Despite the categorization of SoS regarding its form of authority, there are variations. While the four types of SoS presented are quite distinct, SoS has extensive limits, and then different parts of the same SoS can exhibit different types of classification. Moreover, CS may collaborate simultaneously to more than one SoS with different authority structures (NCUBE et al., 2018), as shown in Figure 2.2. The key challenges in this context include specifying SoS requirements, how to predict SoS performance, how SoS requirements affect the CS, how the existing CS limit the requirements for SoS, and how to ensure the SoS accomplishes its missions.

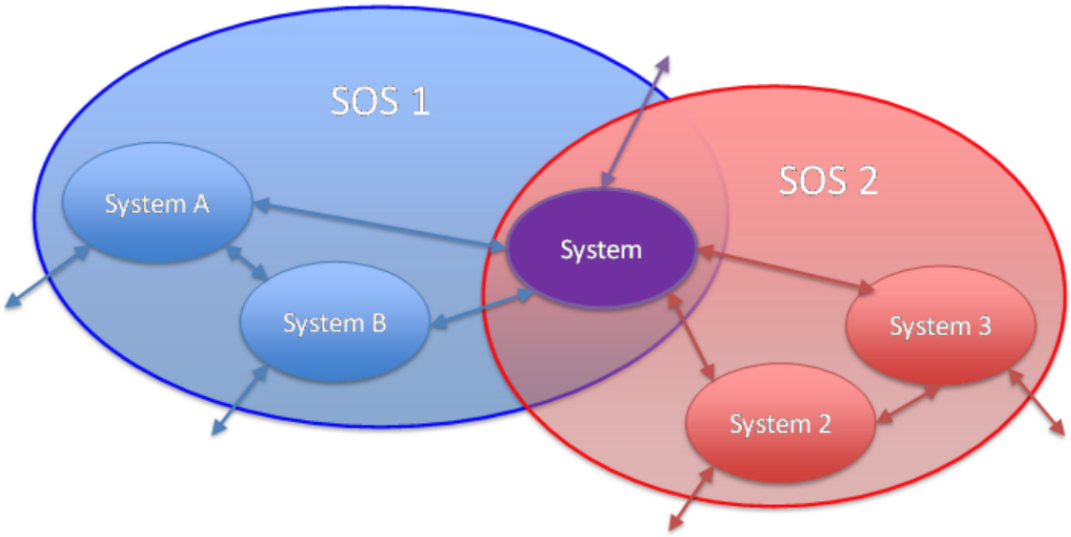


Figure 2.2: Constituent systems can participate in multiple SoS. Source: (NCUBE et al., 2018)

2.2 The mKAOS Language

The mKAOS (SILVA; BATISTA, et al., 2015b) is a mission-oriented modeling language that was created specifically to represent how CS interoperate in an SoS to accomplish missions. It is based on the conceptual model proposed by Silva et al. (SILVA; CAVALCANTE, et al., 2014), who identified the elements of an SoS mission. The mKAOS is a specialization of KAOS (Keep All Objectives Satisfied) (VAN LAMSWEERDE, 2001), a requirements engineering methodology used in the

industry that defines goals as its fundamental element. Figure 2.3 shows the conceptual model for SoS missions. The heuristics can be understood as constraints applied to the missions.

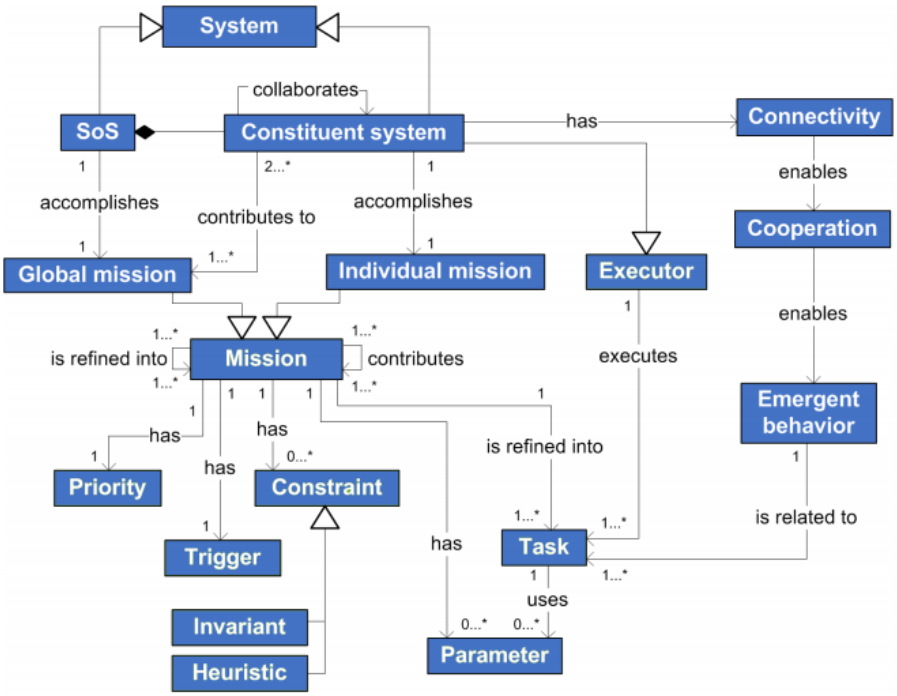


Figure 2.3: Conceptual model for missions of SoS (SILVA; BATISTA, et al., 2015a).

mKAOS has structurated in six different models to represent SoS elements of the conceptual model for missions introduced in Figure 2.3. The main model is the **Mission Model** that describes the missions and expectations. The **Responsibility Model** addresses issues related to the assignments of missions and expectations to the respective CS and environmental agents. The **Object Model** specifies the entities and events handled by the system, the domain hypotheses, which are desirable characteristics and constraints, and the domain invariants, which are constraints that must be maintained as SoS operates and evolves.

The **Operational Capability Model** defines the capabilities provided by CS for the SoS. The **Communicational Capability Model** specifies the interactions and cooperation among CS that are part of the SoS. Finally, the **Emergent Behavior Model** describes the behaviors produced by the interaction of two or more CS while the SoS operates. These six mKAOS models work as viewpoints that allow stakeholders to design and analyze SoS (SILVA, 2018).

As a part of mKAOS work, researchers also developed the mKAOS Studio, a graphical tool based on open-source frameworks within the Integrated Development Environment (IDE) Eclipse [3] platform. mKAOS (i) describes the interplay among CS, missions, refinement of capabilities, objects, and emergent behaviors, often ignored in other modeling tools, and (ii) produces detailed models that architects, managers, and stakeholders can understand.

2.3 Heuristics

Logic, probability, and heuristics are three essential theories in humanity's intellectual history as problem-solving techniques (GIGERENZER, 2008). Since Aristotle, logic was faced as a philosophy of perfect human thinking and inference, aiming to preserve the truth. Probability Theory emerged in the late 17th century, replacing logical certainty for a more modest theory of rationality, recognizing the fundamental uncertainty of human conduct (DASTON, 1980). From that moment on, Probability Theory has transformed science and people's lives, providing statistical methods used in scientific experimentation and solutions to everyday problems, providing data with a degree of certainty that can be calculated.

As a third way to facilitate decisions, heuristics can be defined as mental operations that are employed in the judgment under uncertainty (TVERSKY et al., 1974). Unlike statistical methods, heuristic is a strategy that ignores part of the information, with the goal of making decisions more quickly, frugally, and/or accurately, not doing optimization to find the best solution but instead find a "good-enough" satisfying answer (GIGERENZER; GAISSMAIER, 2011). Calculating a function with the maximum precision is a form of optimizing while choosing the first option that exceeds an aspiration level is a form of satisficing.

Since the early twentieth century, heuristics have been used in a variety of fields, including psychology, where cognitive heuristics are studied in human decision making under uncertain conditions (SHERMAN et al., 1984). In political science, moral heuristics are observed to play a pervasive role in moral, political, and legal judgments (SUNSTEIN, 2005). In usability, heuristics are used to find usability problems in user interface design evaluation (NIELSEN, 1994). Table 2.1 shows the set of ten

[3] https://www.eclipse.org/

Nielsen heuristics for user interface evaluation.

Table 2.1: Usability heuristics for user interface. Adapted from (NIELSEN, 2005).

#	**Issue**	**Statement**
1	Visibility of system status	Designs should keep users informed about what is going on, through appropriate, timely feedback.
2	Match between system and the real world	The design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon.
3	User control and freedom	Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action.
4	Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5	Error prevention	Good error messages are important, but the best designs carefully prevent problems from occurring in the first place.
6	Recognition rather than recall	Minimize the user's memory load by making elements, actions, and options visible. Avoid making users remember information.
7	Flexibility and efficiency of use	Shortcuts — hidden from novice users — may speed up the interaction for the expert user.
8	Aesthetic and minimalist design	Interfaces should not contain information which is irrelevant. Every extra unit of information in an interface competes with the relevant units of information.
9	Help users recognize, diagnose, and recover from errors	Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.
10	Help and documentation	It is best if the design does not need any additional explanation. However, it may be necessary to provide documentation to help users complete their tasks.

In software engineering, heuristics have been used in several contexts, including design patterns, describing groups problems, and helping designers find and correct them. In addition, heuristics can enable a 'softer' model to be constructed to obtain a more holistic and subjective view of quality (CHURCHER et al., 2007). The heuristics proposed by Arthur Riel are an excellent example that cover aspects of object-oriented software development, including classes and objects, multiple inheritances, and association relationship (RIEL, 1996). Table 2.2 shows the heuristics proposed by Riel to address classes and objects.

Heuristics proposed by Nielsen (NIELSEN, 1994), aiming to facilitate the user interface design process, and those proposed by Riel (RIEL, 1996) that help to avoid dangerous decisions in object-oriented software development, can be used as guidelines to help software designers without however providing perfect solutions.

Table 2.2: Classes and objects heuristics. Adapted from (RIEL, 1996).

#	**Statement**
2.1	All data should be hidden within its class.
2.2	Users of a class must be dependent on its public interface, but a class should not be dependent on its users.
2.3	Minimize the number of messages in the protocol of a class.
2.4	Implement a minimal public interface that all classes understand.
2.5	Do not put implementation details such as common-code private functions into the public interface of a class.
2.6	Do not clutter the public interface of a class with items that users of that class are not able to use or are not interested in using.
2.7	Classes should only exhibit nil or export coupling with other classes, that is, a class should only use operations in the public interface of another class or have nothing to do with that class.
2.8	A class should capture one and only one key abstraction.
2.9	Keep related data and behavior in one place.
2.10	Spin off non-related behavior into another class (i.e., non-communicating behavior).
2.11	Be sure the abstractions that you model are classes and not simply the roles objects play.

2.4 Final Remarks

This chapter brought up concepts about SoS, presenting the types and characteristics of this kind of system, about the mKAOS, a modeling language developed specifically to model SoS and their missions, and about heuristics proposed for use in interface design and object-oriented development, intended to help in software design.

In the Chapter 3, an exploratory study is presented in order to understand better how the SoS approach is seen in the industry. This work was performed by surveying practitioners in a real-world public organization about an SoS characterization using the mKAOS mission model as well as the problems and concerns perceived in the characterized SoS.

Chapter 3. Exploratory Study

In this chapter, an exploratory study is presented. It was conducted to verify problems and concerns related to an SoS running in a real-world environment. Section 3.1 presents an introduction to the study. In Section 3.2, there is a characterization of the organization in which the SoS is in operation. Section 3.3 presents the investigated SoS problems. Section 3.4 proposes a modeling extension to represent SoS problems. Section 3.5 presents the study limitations, and Section 3.6 the final remarks.

3.1 Introduction

A study about SoS was carried out in a Brazilian public organization to learn more about how an SoS works and how problems and issues takes place in a real-world organization. This study aimed at appropriating SoS concepts and realizing what kind of issues could be foreseen and dealt with in an SoS using heuristics yet at design time.

An SoS characterization was proposed using information from interviews with professionals and by reading systems documentation. This allowed us to identify the SoS missions, CS, constraints, relationships, and capabilities. We modeled the SoS using the mKAOS language to support the characterization of the SoS.

Using the SoS model produced as a basis, a focus group and a survey were conducted with professionals involved in SoS engineering, aiming to identify situations that could affect the SoS operation and how they could be addressed. The focus group and the survey contributed to the proposal of an initial extension to the mKAOS notation that allows representing issues observed, such as failures and fault tolerance mechanisms in SoS.

3.2 System-of-systems of a Brazilian Public Organization

This study was conducted in a Brazilian public organization with head office in Rio de Janeiro, regional offices in 27 states, and more than 7,000 employees. The regional offices coordinate 564 branches distributed across the country. The branches are responsible for collecting demographic data through questionnaires applied during face-to-face visits, applications via the Internet, or phone calls.

The SoS in operation in this organization provides data fusion capabilities that support decision-making processes. Such a capability depends on data from different systems, which we divided into two groups: technical and administrative. The technical group involves systems that collect, store and process demographic and economic data and produce statistics. The administrative group consists of systems that support administrative processes, including human resources and contracts. The systems were developed or acquired separately, which implies that different teams are responsible for each IS, and their evolution reflects their specific demands. The infrastructure that supports each IS is diverse in terms of technological platforms, such as databases and programming languages.

Technical CS store demographic and economic data holds information on urban infrastructure (i.e., healthcare, education, mobility, buildings etc.), agricultural activity, economic activity, population profile, and employment. Administrative CS support the management of human resources, financial and patrimonial data. In addition, administrative CS store information about operational activities, such as vehicle utilization and travel expenses. Table 3.1 lists systems and describes the data that each of them provide to the SoS. Obvious names were adopted for the sake of simplicity.

Table 3.2 describes the SoS and CS missions. M.02 mission consists of consolidating the work done and the location of the country's demographic and economic data collection activities of the M.05 to M.09 missions. It contains information about the total work done with collected data. The available workforce mission M.03 refers to the sum of the number of temporary and permanent employees. Mission M.04 is the total business travels made by the employees using flight tickets (M.10) or vehicles to transport employees (M.11). The primary mission M.01 consists of the relation between the work done (M.02) versus the resources used (M.03 and M.04).

Table 3.1: Summary of constituent systems.

ID	Constituent system	Group	Information produced	Data provided to SoS
CS.01	[Cities]	Demographics and economic	Urban infrastructure, such as healthcare, transportation, and education.	Number of inquiries.
CS.02	[Population]	Demographic and economic	Deaths, births, and life expectancy.	Number of inquiries.
CS.03	[Employment]	Demographic and economic	Employment rate and household income.	Number of inquiries.
CS.04	[Economic Activity]	Demographic and economic	Employment and income in economic data for business.	Number of inquiries.
CS.05	[Agriculture]	Demographic and economic	Agricultural production.	Number of inquiries.
CS.06	[Temporary Workers]	Administrative	Details on temporary employees, such as location, job description, and contracts.	Available temporary workers.
CS.07	[Permanent Workers]	Administrative	Details on permanent workers, such as vacation, job description, and location.	Available permanent workers.
CS.08	[Transportation]	Administrative	Details on work-related travels, such as destinations, use of company vehicles, flight tickets etc.	Travel destinations and resources used.

Although this SoS can be considered a directed SoS, conflicting situations in which accomplishing the SoS mission are not a priority to any particular CS manager. The SoS mission may not be treated as a priority by any CS as it is not part of their individual objectives.

Table 3.2: SoS and CS Missions.

ID	Description	Type
M.01	Expense control	SoS mission
M.02	Demographic and economic profile	SoS mission
M.03	Available workforce	CS mission
M.04	Total of travels	SoS mission
M.05	Urban statistics	CS mission
M.06	Population statistics	CS mission
M.07	Employment and income household statistics	CS mission
M.08	Economic statistics	CS mission
M.09	Agricultural statistics	CS mission
M.10	Travels' registration: costs, dates, and passengers	CS mission
M.11	Vehicles' utilization: costs, dates, and passengers	CS mission

To support the investigation, a model was constructed following the mKAOS mission language. This model aimed to verify with practitioners if this arrangement of systems fulfilling new missions could be characterized as an SoS. Mission model

is one of the six viewpoints implemented in mKAOS language (SILVA; BATISTA, et al., 2015b).

mKAOS mission model (i) describes the interplay among CS, missions, refinement of capabilities, objects, and emergent behaviors, and (ii) produces detailed models that architects, managers, and stakeholders can understand. The elements of the mKAOS notation used in this work and the first generated model are shown in Figure 3.1.

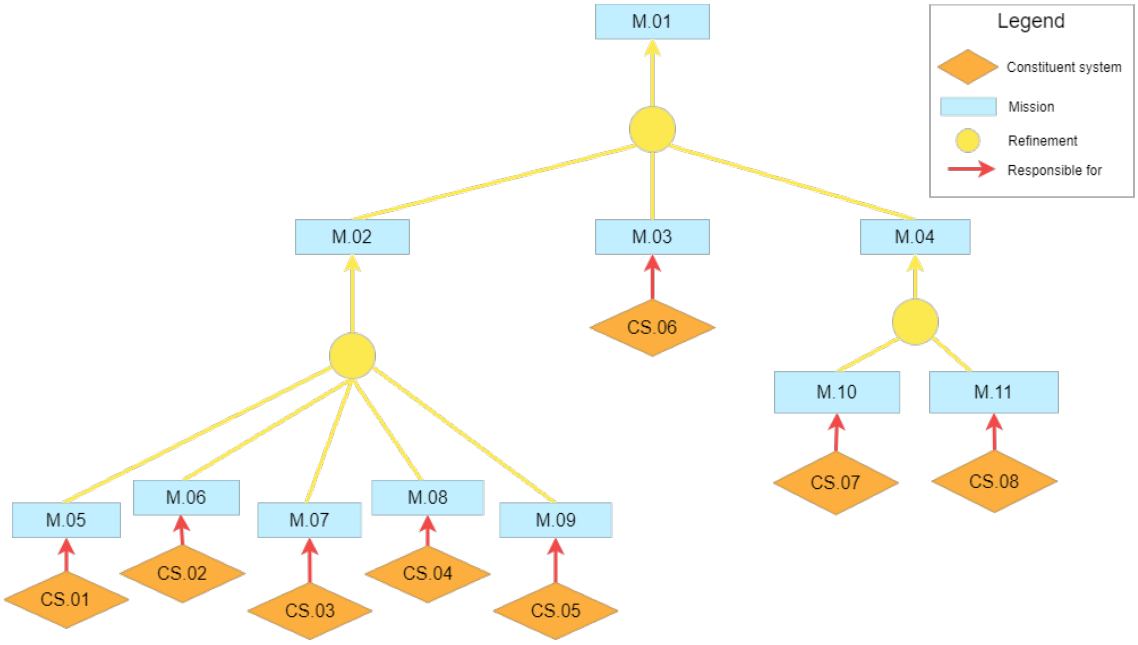


Figure 3.1: mKAOS mission model.

3.3 Investigating SoS problems

Two research questions Were formulated to guide our investigation:

RQ1. Which are the problems regarding operation of the SoS?

RQ2. How can those problems be represented in the mKAOS mission model?

Two studies were conducted to investigate the SoS problems and whether mKAOS would be suitable for representing them: a focus group and a survey. The focus group aimed to openly discuss how CS work to fulfill SoS missions and which concerns related to reliability should be considered. With the information obtained, an mKAOS mission model was generated with some annotations representing possible

points of failure in SoS and forms of recovery. This model was then presented to professionals in a survey to verify the feasibility of incorporating new elements that allow the problems representation into the mKAOS mission model.

3.3.1 Which are the problems regarding SoS operation?

A focus group was planned and executed to discuss the SoS operation and identify situations that could jeopardize such operation. The focus group had five participants: a senior manager and two senior developers from the organization, and two researchers, one with an Master's degree and an M.Sc student. Before the meeting, a summary of the material produced during an initial SoS modeling was provided to the participants. The material consisted of (i) the definition of SoS and its characteristics according to the architectural SoS principles (MAIER, 1998); (ii) basic concepts of quality attributes; (iii) the SoS mKAOS mission model produced (Figure 3.1); and (iv) a brief explanation of some potential problem situations that could be identified during the SoS characterization.

The focus group had the following agenda: (i) presentation of the participants, (ii) clarification on the definitions of SoS and quality, (iii) evaluation of concepts and the SoS model, (iv) verification of the usefulness of the model to address SoS quality, and (v) comments and suggestions. During the discussions, one participant emphasized that he had little familiarity with SoS concepts. Despite that, the model was considered sufficient for all participants to recognize the arrangement of independent systems as an SoS and how each CS contributes to accomplishing the overall mission.

The participants identified some problems as failures that were classified as temporary and permanent. A failure is considered temporary when it occurs for some time and disappears with no intervention. A failure is considered permanent when it is continuous. In this case, a repair or component replacement is required. For instance, an interruption in the power supply is a temporary failure for someone using a computer, while a broken keyboard is a permanent failure. Changes in the implementation of a CS imply permanent failures for the SoS. For instance, changes in the parameters for requesting a service or changing the data format.

Participants discussed two types of temporary failures that imply data unavail-

ability: network failures and data service failures. Network failures may occur due to high network usage or connectivity interruptions, which causes delays and/or packet loss. Data service failures can occur due to an excessive number of calls on web services. The failures identified are described in Table 3.3.

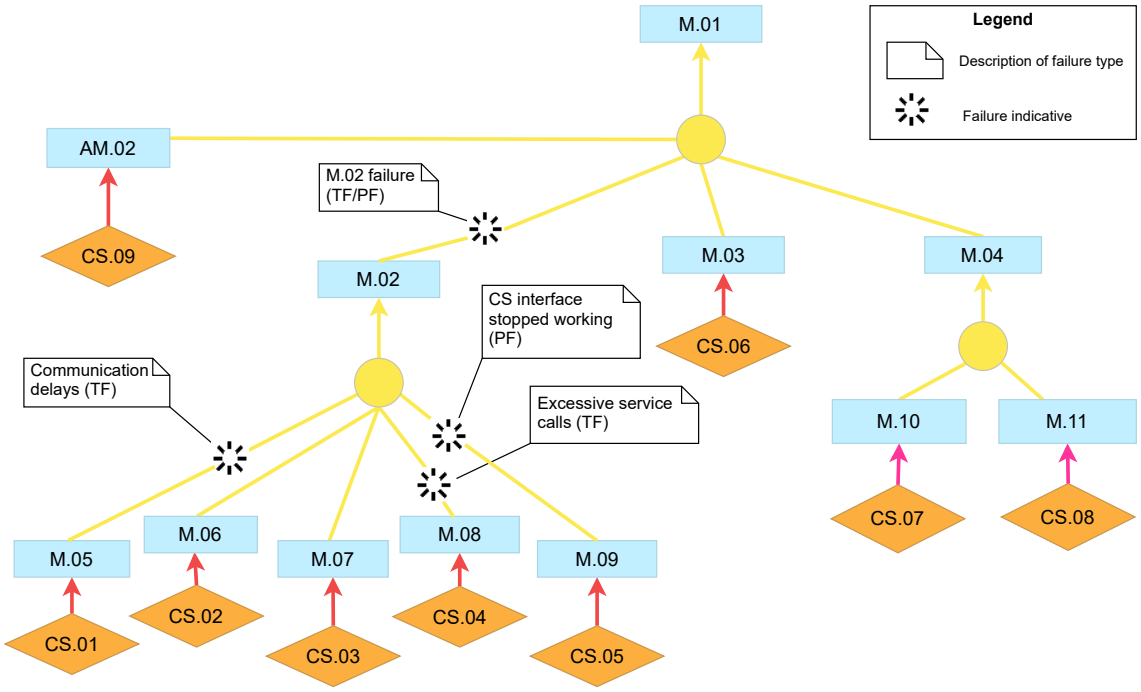


Figure 3.2: mKAOS mission model annotated for the evaluation.

Table 3.3: Failures identified in the SoS.

Failure	**Type**	**Description**
Communication delays	Temporary	Data that depend on manual collection is sent in batches to the respective IS and can take up to a month to be incorporated in the database.
Service unavailability	Temporary	Excessive calls to data services (microservices, databases etc.) and network traffic cause services unavailability.
CS interfaces have changed without notice	Permanent	Unannounced maintenance of a given CS causes interruptions in data provision. Examples of effects of such failures are unresponsive services and unknown data formats.

As the federal government imposes the adoption of practices that assure transparency policies, a significant portion of the previous years' institutional data is also available in open data portals. Such data are an alternative source of information when the primary sources coming from CS are unavailable.

Participants reported strong reluctance of managers in sharing information among the organizational sectors. This situation implies difficulties in building a strategic

vision of the whole organization. The lack of synergy among all the CS affects the long-term sustainability of the SoS. Problems that do not directly affect the operation of a CS individually - such as interoperability problems - can make the SoS inoperable or unavailable. In these cases, the SoS capability is recovered when an authority requests it.

3.3.2 How can problems be represented in SoS?

The focus group shed light on mainly in reliability issues. The circumstances described by the participants served as a basis to elaborate a survey for further exploring reliability representation in the mKAOS mission model. A new material was developed on the SoS mission model with annotations that were presented to the professionals. The material included a modified version (Figure 3.2) of the SoS mission model with failure indicative symbols that points to failures and one potential redundancy solution, the *Open Data Portal* (CS.09). The Open Data Portal can support SoS in accomplishing the mission since it has capabilities similar to both the *Number of Travels* mission and the *Demographic and Economic Profile* mission, but providing only annual historical data, so there is a significant delay in updating data with this redundant CS capability.

The material presented to the respondents were (i) the terms and concepts of SoS and reliability and (ii) the modified SoS mission model with the propositions perceived in the focus group discussions. Figure 3.3 shows an example of a possible failure. The survey's objective was to verify if the annotated mKAOS mission model was easy to understand and if it helped to properly visualize circumstances that could affect the SoS. A five-point Likert scale was used for all survey questions (1. Strongly disagree; 2. Disagree; 3. Indifferent; 4. Agree; 5. Strongly agree). An invitation was sent to 20 professionals from the CS engineering teams, including managers, developers, and architects — all of them with expertise on the different CS that compose the SoS. The survey was answered by 11 participants. One of them holds a Ph.D degree, 3 hold a Masters' degree, 4 hold a Specialization degree, 2 hold a Bachelor's degree, and one participant completed high school.

The survey questions are described in Table 3.4 and a summary of the answers is shown in Figure 3.4. Q.01 and Q.02 answers reinforce the model's usefulness

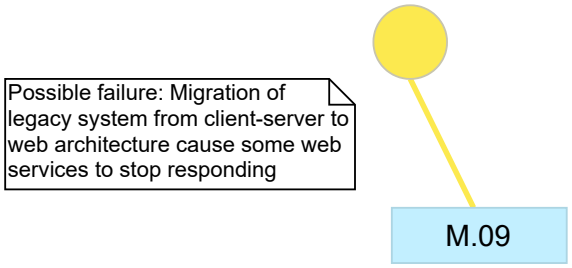


Figure 3.3: Part of the material presented in the survey.

in describing how CS are organized to fulfill the SoS mission. In Q.03, it was observed that the failure indicative symbols with text box descriptions representing redundancy situations were accepted as a resource to describe the reliability issues, although they are just annotations above the model.

Table 3.4: Survey questions.

ID	**Question**
Q.01	Does the model allow an understanding of SoS objectives?
Q.02	Does the model describe how the different CS contribute to the SoS?
Q.03	Is the model capable of describing any redundancy in the case of failures of any CS?
Q.04	Is the model capable of describing situations in which the SoS recovers from failures?
Q.05	Does the model allow the understanding of situations that affect availability (i.e., the system's ability to be available when required)?

Functional redundancy consists of using two or more different components to accomplish a given task (JACKSON et al., 2013). One respondent commented that it is not clear how redundant elements are activated in case of CS unavailability. Analyzing the Q.04 answers was perceived that few participants identified how the SoS recovers from failures. Q.05 answers reveal that it was not clear how to check availability by looking at the mKAOS model, even with the provided annotations. In addition, the responses to Q.01 and Q.02 show that the SoS concepts were well understood, and the mKAOS model was useful to represent the SoS. However, responses to Q.03, Q.04, and Q.05 show that, even with the verbatim representation of the failures written over the model, the reliability issues could not be understood right away reading annotations.

Regarding the failures identified in the study, they can occur due to several causes, including changes in implementation, malfunction, bugs, hardware problems etc. Communication channels are significant sources of problems. SoS failures can

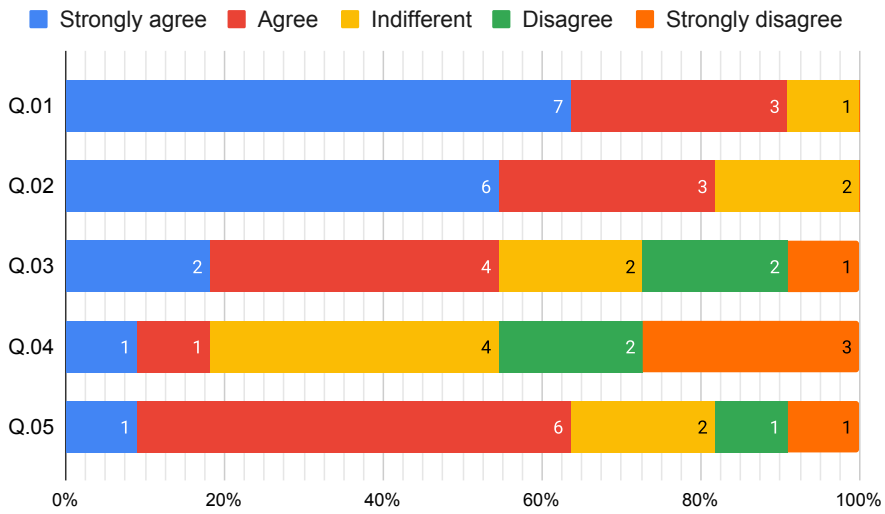


Figure 3.4: Summary of the survey answers.

occur even if none of the CS have any malfunction nor fail to meet any specifications for which it was designed. In other words, a reliability issue in the SoS is not necessarily a problem for any of the CS.

Even though the mKAOS model allows the understanding of the CS interplay in accomplishing missions, a new set of elements is necessary to represent failure situations and how these problems can be avoided in interpreting the model. Non-functional requirements such as reliability are not covered by mKAOS notation, requiring additional artifacts to address such aspects. Therefore, an mKAOS extension was proposed to represent the situations identified in the studies conducted in a real world SoS.

3.4 mKAOS extension proposal

After the investigation, it was possible to propose an extension to the mKAOS notation to represent reliability issues in mKAOS mission model. The extension was inspired by the Business Process Model and Notation (BPMN), largely recognized in the industry.

The BPMN's primary objective is to provide a unique notation of business process (OMG et al., 2011). The use of elements already known in modeling languages can facilitate understanding and standardize communication. Some of these symbols could be applied to describe redundancy mechanisms and failure situations in

an SoS. Figure 3.5 shows the BPMN symbols used to represent such situations. The BPMN events symbols are applied to describe failures that affect CS capabilities provision, while gateway symbols are applied to represent how these failures are addressed.

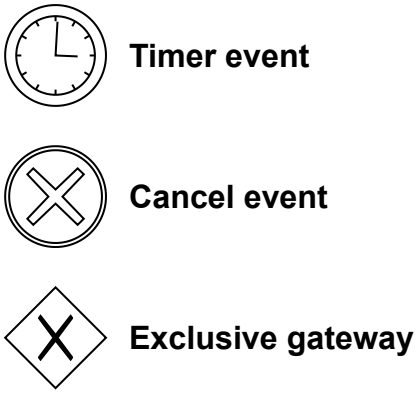


Figure 3.5: Symbols from BPMN used in this work.

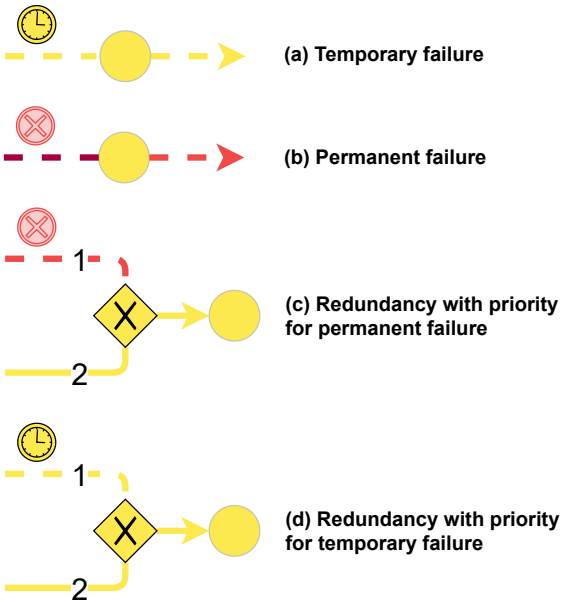


Figure 3.6: Elements for reliability in the mkAOS extension.

The "Exclusive Gateway" symbol represents situations in which there are two or more redundant capabilities available in a given SoS. Only one of them should be selected as the priority option. Temporary failures were represented with the "Timer" symbol, while permanent failures were represented with the "Cancel" symbol. Figure 3.6 illustrates the new elements proposed to compose the mKAOS extension with the representation of reliability aspects.

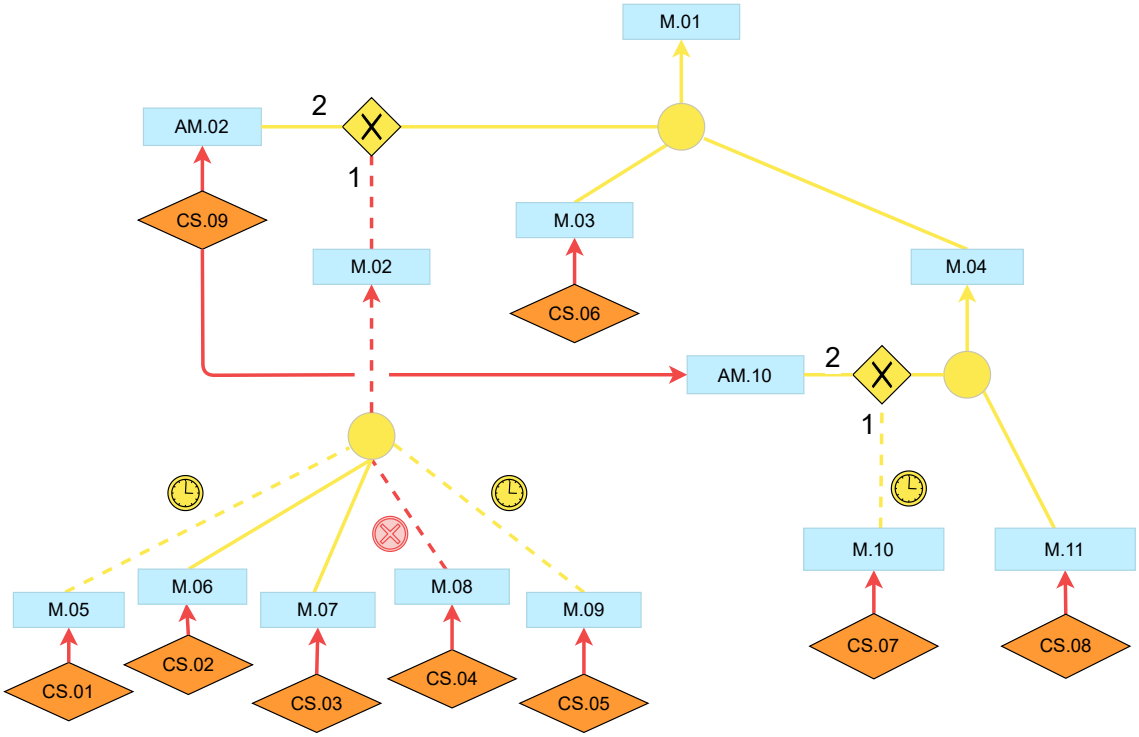


Figure 3.7: mKAOS mission model with the proposed extension.

A representation of a temporary failure is shown in Figure 3.6.a. The symbol is positioned on edge directly linked to the capability provided to represent the temporary failure. This notation focuses on the edge because the failure itself is not necessarily a CS failure but a failure to provide a capability to an SoS. The dotted yellow edges denote that the capability provided by a given CS can be interrupted, and these dotted edges are used in all the lines above after the respective refinement until the failure is properly addressed. Figure 3.6.b illustrates a permanent failure. The representation is positioned at the edge directly linked to the respective capability.

Figure 3.6.c describes the element representing redundancy. The priority for activation of each capability is also indicated to facilitate using the model to communicate which CS is immediately activated in case of failures. The same applies to redundancy for temporary failures, as shown in Figure 3.6.d. The redundancy sources connected to the gateways address the problems caused by temporary or permanent failures and spread the effect to subsequent higher nodes, making the dotted lines continuous.

Figure 3.7 shows the model with possible failures and how they are addressed.

The redundancy mitigates the situations of a permanent failure in M.02 and temporary failure in M.10, implying activation of AM.02 and AM.10 mission to provide data from previous years as an alternative to M.02 and M.10, respectively. Delays in data transmission from M.05, M.09, and M.10, represented with temporary failure symbols, may be acceptable to a CS and negatively affect the SoS mission. A permanent failure in M.08 occurred when CS.04 technological platform was upgraded and changed its communication protocol. Redundancy is activated by using CS.09 (Open Data Portal) to provide a redundant capability for M.10.

3.5 Limitations

The focus group has advantages and disadvantages regarding the number of participants. Small groups, such as in this work, make discussions more comfortable, enhance participants' involvement, provide more time for individual participation, and moderate conflicting opinions properly. On the other hand, large groups facilitate the composition of a broader vision on the subjects (KITZINGER, 1994).

Redundancy is one of the mechanisms of fault tolerance, one of the four characteristics that software must have to be considered reliable, according to ISO 25010. Hence, further studies are necessary to represent other reliability issues. Moreover, in the studied scenario, only a directed SoS was investigated. Different levels of managerial control and authority over the CS imply that different SoSE approaches may be required in other types of SoS. Hence, collaborative, acknowledged, and virtual SoS types need to be addressed in further researches.

Although the number of people in the survey can be considered acceptable (NULTY, 2008), the results cannot be generalized since a single case has been studied. In addition to this real SoS running in a Brazilian public organization, other SoS cases need to be investigated in different public and private institutions to enrich results with more views on the subject and contribute with more in-depth investigation and results. Finally, before the initial mKAOS extension proposal can help SoS engineers to deal with reliability issues, it is necessary to properly evaluate and adjust it before considering the feasibility of incorporating it into the mKAOS tool.

3.6 Final Remarks

Problems including reliability are non-functional requirements of IS widely considered in quality assessment and well defined by well-established norms and standards worldwide. However, there are gaps to be filled when moving to the SoS context. Reliability is a fundamental attribute to be treated in SoS since the CS independence, and the resulting dynamic architecture of the SoS can bring new reliability issues not observed in CS individually. Although SoS reliability is not yet a widely addressed problem, some studies describe these problems' source and propose practices to minimize failures. In addition, recognizing the different types of failures is essential in deciding the most appropriate means to adequately address each type of problem and mitigate the risks for accomplishing SoS missions.

The model of the SoS in operation in a Brazilian public organization developed from the mKAOS language was useful to allow the participants to produce valuable discussions in the focus group. It was possible to describe failures and fault tolerance mechanisms. The description of reliability issues using a model facilitates communication and helps to achieve solutions to deal with reliability issues.

An evaluation of the extension was conducted to investigate its usefulness and potential improvements. After the extension of mKAOS evaluation and adjustements, a tool was developed aiming to help professionals deal with SoS operation issues (IMAMURA; FERREIRA; SANTOS, 2020). Such a tool could be helpful to analyze the CS attributes and how their interplay within an SoS, allowing the reliability issues and other issues to be identified and solved at design time.

This study used only the mKAOS mission model notation and a proposed extension. As future work, the SoS representation could be enhanced using other viewpoints from mKAOS notation so that more SoS components and features can be represented, improving the understanding of SoS-related issues.

The case presented in this work is not exhaustive. Despite that, it was expected that this work allows a reflection on the adoption of practices to recognize, eliminate, reduce, or avoid conditions that may degrade SoS operation.

This exploratory study was conducted to understand how an SoS operates in the real world and what the concerns are seen by professionals working with CS involved in the SoS operation. It was observed that there are issues that negatively affect the

SoS, but are not perceived by any of the CS. It might be useful to know what are those new issues and how they can be handled at the SoS level. One way to deal with those issues is to find heuristics that can be applied to handle such problems in SoS, even in design time.

In Chapter 4, a research methodology was defined and executed aimed to investigate what are the heuristics suitable for SoS and how they can be organized and applied in the design phase.

Chapter 4. Heuristics Catalog for SoS Design

The objective of this chapter is to describe the process of extraction, organization and evaluation of the heuristics for SoS design. Section 4.1 presents an introduction. In Section 4.2, we present a systematic mapping study, aimed to extract heuristics applied do SoS design from literature. Section 4.3 presents a focus group conducted to organize the heuristics. Section 4.4 describes a survey conducted to evaluate the organized heuristics. Section 4.5 describes the limitations of this study and Section 4.6 present the final remarks.

4.1 Introduction

To construct the proposed heuristics catalog, a SMS was selected as the method to extract the heuristics from literature. A total of 40 heuristics were extracted from selected studies from different authors, so there are similar, complementary, and divergent views on how heuristics can be applied in SoS design.

Additional work should be done to organize the extracted heuristics from SMS, making them useful in practical SoS design activities. A focus group was conducted to refine and organize the initial set of 40 heuristics. The focus group was selected as the appropriate instrument to capture the perception of experts and refine the ideas and concepts extracted from the literature.

After the refinement done in the focus group analysis, a survey was conducted with SoS experts to evaluate the resulted set of heuristics.

4.2 Systematic Mapping Study

The design of an SoS presumes that the collaboration among independent systems can help solve a problem that could not be solved by a single system alone (SAGE et al., 2001). However, ensuring SoS quality is particularly challenging since it is necessary to deal with the intrinsic characteristics of this type of system, which often deals with unpredictability and uncertainties.

To deal with these issues can be difficult. For instance, is there any way to monitor or mitigate undesirable behaviors when a CS changes the level of its contribution or leaves the SoS without notice? This situation can continually affect the accomplishment of SoS missions, and it is necessary to create solutions to minimize uncertainties when designing SoS.

To understand how these problems have been addressed, we carried out a systematic mapping study. To do so, we adopted the GQM (Goal-Question-Metric) method to define the objective (BASILI et al., 1994): **analyze** the SoS literature **with the purpose of** characterizing the current state-of-the-art **with respect to** rules, design principles, recommendations, and standards for the design of SoS **from the point of view of** researchers **in the context of** systems-of-systems. Based on this objective formulated with GQM support, we proposed two research questions:

RQ1. Which heuristics have been applied to SoS design?

Rationale: Identify design principles, good practices, and recommendations that have been adopted in the design of SoS.

RQ2. For which types of SoS the heuristics can be applied?

Rationale: One of the characteristics to be considered in the SoS design is how the CS are coordinated to provide the required capabilities. The type of SoS can facilitate or hamper the adoption of heuristics. Thus, the purpose is to identify for which types of SoS each heuristic can be applied.

A search string was defined from the keywords “System-of-Systems”, “Heuristics”, “Design Principle”, “Pattern”, and “Rule”. The keywords were connected using the AND logical operator, while variations and synonyms were connected using the OR operator. The terms of the search string were selected aiming at a broader search, i.e., a large coverage of studies. We tested different configurations

of the search string in Scopus[1], which is considered the largest scientific publication database that indexes the most relevant publication venues. After calibrating the search string, the final string was:

```
("system-of-systems" OR "SoS" OR "system of systems" OR "systems
   of systems" OR "systems-of-systems") AND ("heuristics" OR
  "heuristic" OR "design principle" OR "design principles" OR
        "pattern" OR "patterns" OR "rule" OR "rules")
```

We also carried out searches in ACM Digital Library[2] and IEEExplore[3]. At the end of the searches, 3,765 studies were retrieved. Then, we removed the duplicated studies, resulting in a set of 3,645 studies. Next, we looked at the title, abstract, and keywords and applied the selection criteria described in Table 4.1 where IC1 is the inclusion criterion and EC1 to EC5 are the exclusion criteria.

We discarded 3,393 studies and 372 remained for a detailed analysis. Following, we read the abstract and conclusion of the remaining 245 studies and, after applying again the selection criteria, 14 studies were selected for the data extraction. The selected studies are listed in Table 4.2.

Table 4.1: Selection criteria.

ID	**Criteria**
IC1	The study presents a discussion on systems-of-systems heuristics, design principles, patterns or rules.
EC1	Study is not available for download openly or through institutional access and could not be retrieved from the author.
EC2	Study is not written in English.
EC3	The study is a book, tutorial, editorial, abstract, poster, panel, lecture, round table, workshop, demonstration or preface.
EC4	The study is not a primary study.
EC5	Study deals with concepts and acronyms not specifically linked to systems-of-systems.

We adopted the process described in Figure 4.1 as our research method to carry out the systematic mapping study. At the end of the process, 14 studies was selected. Table 4.2 shows the studies with respective reference.

[1]http://www.scopus.com
[2]https://dl.acm.com/
[3]https://ieeexplore.ieee.org/

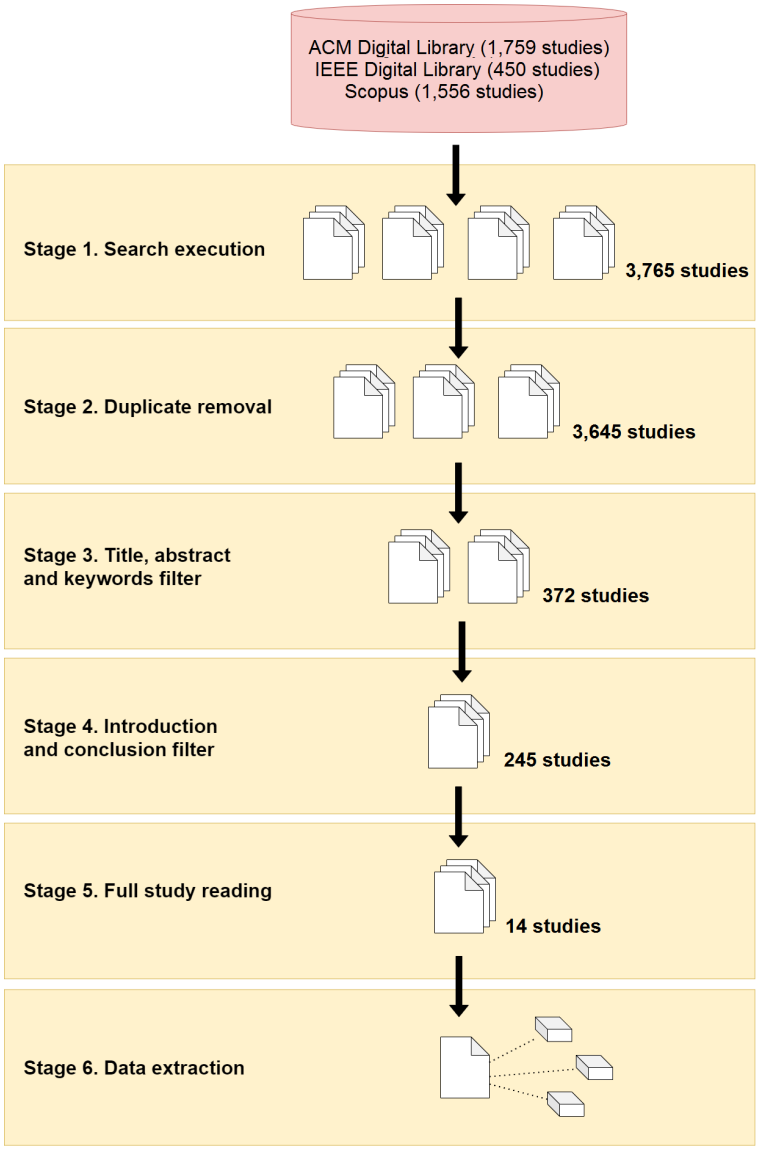


Figure 4.1: Selection process.

4.2.1 Results

Despite the use of the term "heuristic" and its synonyms, the studies S12, S13, and S14 are not considered applicable heuristics according to the proposed formulation previously shown in Table 2.1 and Tables 2.2. Study S12 presents a 5-step roadmap to facilitate the choice of interoperability standards for SoS deployment. The study S13 proposes an 8-phase process to address SoS missions. Study S14 deals with creating test cases, a step to be executed after the SoS design.

In Table 4.3, we present the 40 extracted heuristics and the respective rationales from the studies S01 to S11. Table 4.4 describes the types of SoS addressed and

Table 4.2: Selected studies

ID	Study	Reference
S01	SoS-Centric Middleware Services for Interoperability in Smart Cities Systems	(LOPES et al., 2016)
S02	Developing systems thinking skills using healthcare as a case study	(MCDERMOTT, 2018)
S03	Architecting principles for systems-of-systems	(MAIER, 1998)
S04	Challenges for SoS Architecture Description	(BATISTA, 2013)
S05	Towards a Dynamic Infrastructure for Playing with Systems of Systems	(SCHNEIDER et al., 2014)
S06	When Ecosystems Collide: Making Systems of Systems Work	(SILVA AMORIM et al., 2014)
S07	A cooperative SoS architecting approach based on adaptive multi-agent systems	(BOUZIAT et al., 2018)
S08	A generalized options-based approach to mitigate perturbations in a maritime security system-of-systems	(RICCI et al., 2013)
S09	A Meta-Process to Construct Software Architectures for System of Systems	(GONÇALVES et al., 2015)
S10	Harnessing Emergence: The Control and Design of Emergent Behavior in System of Systems Engineering	(MITTAL et al., 2015)
S11	On the Challenges of Self-Adaptation in Systems of Systems	(WEYNS et al., 2013)
S12	Randomisation in designing software tests for systems of systems	(LIANG et al., 2012)
S13	"Understanding Patterns for System of Systems Integration	(KAZMAN et al., 2013)
S14	A Process to Establish, Model and Validate Missions of Systems-of-Systems in Reference Architectures	(GARCÉS et al., 2017)

how each of the heuristics was evaluated by the primary studies.

Table 4.3: Extracted heuristics.

ID	Ref.	Description	Rationale
H01	S01	The SoS coordination must ensure that each CS can exchange and understand data and messages with the other CS.	Communication standards need to be maintained by the SoS coordination to ensure that each constituency can communicate with the others when necessary.
H02	S01	The SoS coordination must ensure that the policies for accessing and using the capabilities of the CS can be shared and understood by all CS.	It is necessary to ensure that each CS can participate in SoS respecting other CS' policies. The management and dissemination of these policies are necessary given the characteristics of independence of the CS and the SoS dynamic architecture.

Continue on next page

Table 4.3 – continued from previous page

ID	Ref.	Description	Rationale
H03	S02	The design process must identify who benefits from SoS.	Every SoS is built and operates with a purpose. It is important to identify those who benefit from the activities and/or activities of SoS.
H04	S02	The design process must identify who pays for the development and operation of SoS.	It is necessary to inform the stakeholders how and by whom SoS construction and operation activities will be funded to deal with these issues correctly.
H05	S02	The design process must identify who provides the necessary capabilities and resources for SoS operation.	SoS depends on a synergy between systems that provide the capabilities and resources necessary for its operation.
H06	S02	The design process must identify which stakeholders are disadvantaged when participating in SoS.	It is not always possible to guarantee that all SoS participants can only benefit from being part of the system. Therefore, it is necessary to manage the disadvantages of participating in SoS to ensure CS' encouragement and/or replacement.
H07	S02	The value of a product or service provided by SoS for each participant must be determined.	The value-centered philosophy of the products or services provided by SoS should be used so that efforts are focused on the benefits brought by the results and not on the results themselves.
H08	S02	The participants' perception of how SoS is executed and what the results are must be identified.	The participants' perception of what occurs in SoS and how it delivers the results is more important than the facts on the operation and the results themselves. Therefore, it is necessary to manage the information flows to understand and appreciate how SoS is managed.
H09	S02	Open architectures should be adopted for SoS development and evolution.	The SoS must evolve to adapt to its consumers' needs. Thus, open architectures facilitate the SoS SoS evolution. On the other hand, closed architectures that require the use of proprietary technologies or some licensing can make the SoS evolution difficult or unpractical.

Continue on next page

Table 4.3 – continued from previous page

ID	Ref.	Description	Rationale
H10	S03	Stable intermediate forms of architecture must be defined in the SoS evolution process.	For an SoS to dynamically evolve yet maintaining continuous operation, it is necessary to think of intermediate states to enable the feasible transition.
H11	S03	It must be defined which CS should be helped, which ones can recover on their own and which ones are not worth investing efforts to recover.	Only realistic and possibly practical maintenance strategies should be applied for the maintenance of CS.
H12	S03	Interfaces between the CS must be defined already at design time.	Interfaces between CS are a crucial factor in the operation of SoS as they enable collaboration.
H13	S03	Incentive mechanisms for collaboration among CS must be identified in the design phase.	Collaboration among independent systems may have no apparent use for participants.
H14	S04	The components represented in the model at both SoS-level and CS-level must describe interfaces, properties, and constraints declared unambiguously.	It is necessary to define interfaces, properties, and constraints for the SoS and CS levels for ways to address how individual properties impact SoS reliability.
H15	S05	Interoperability must be assured rather than integration.	Unlike integration, interoperability works with loose coupling among systems, favoring the replacement of CS and the evolution of SoS.
H16	S05	Define which functions should be implemented and which are already available in the CS for developing the SoS.	The design process must anticipate whether there are functionalities available in the CS to collaborate or if this functionality needs to be developed.
H17	S05	Define the data that should be exchanged among subsystems.	The specification of the data set exchanged among the CS must be described in detail to generate the necessary capabilities for an SoS to fulfill its mission.
H18	S05	It is necessary to justify the reason to CS to exchange data.	All CS must be aware of why data is exchanged to ensure value delivery and avoid conflicts.

Continue on next page

Table 4.3 – continued from previous page

ID	Ref.	Description	Rationale
H19	S06	Interface patterns resulting from the evolution must be identified.	The evolutionary process can generate the need to adopt new communication patterns or to update existing patterns. It is necessary to maintain the standards as the CS evolve.
H20	S06	Interfaces must be treated in layers or as service buses.	Layered or service bus design allows replacing modules or studying alternatives for required capabilities without affecting SoS operation.
H21	S06	Validation and verification activities must be applied to all phases of development.	As with traditional systems, rather than testing validation and verification at the end of the lifecycle, each development phase should be followed immediately with these activities that provide feedback so errors can be fixed as soon as possible.
H22	S07	Criticality must be considered for SoS evolution.	Criticality takes into account the dynamics among the CS and the environment. Each CS must autonomously choose which interaction produces less criticality.
H23	S08	Perturbations must be identified to assess architectural alternatives.	It is necessary to understand which alternatives produce the least disturbance during SoS operation. Alternatives must be evaluated with metrics that make it possible to evaluate the solution.
H24	S09	The self-regulatory capabilities of each CS must be identified.	Each CS has the autonomy to regulate itself at a certain level. This autonomy must be represented in the SoS design.
H25	S09	The SoS architecture must allow feedback.	It is necessary to monitor SoS operation to detect problems during its operation. Monitoring can reveal the behavior of the CS and the dynamism of SoS.
H26	S09	The integration of self-managed systems must be consistent with the processes and individual interests of the CS.	How SoS can provide feedback to those responsible for its operation must be foreseen from the design.

Continue on next page

Table 4.3 – continued from previous page

ID	Ref.	Description	Rationale
H27	S09	Support for connectivity among geographically dispersed CS must be provided.	It is necessary to ensure that autonomously managed systems consistently contribute to the processes and interests of other systems in SoS.
H28	S09	Support for connectivity among heterogeneous environments must be provided.	How connectivity among geographically distributed systems occurs must be identified both for the means of communication and the protocols used.
H29	S09	Emergent behaviors must be represented in capability composition schemes.	It is desired that the emergent behaviors are known regarding the capabilities required by the CS and their relationships.
H30	S09	Individual capabilities must be validated.	Each CS provides a capability to SoS that must be checked to see if it complies with SoS expectations.
H31	S09	The design must anticipate desired emergent behaviors.	Emerging behaviors necessary to fulfill missions must be explicitly identified, allocated to the respective CS responsible for them, and evaluated frequently.
H32	S09	Emerging behaviors must be assigned to CS capabilities.	Emerging behaviors necessary to fulfill SoS missions must be explicitly identified, assigned to the respective CS responsible for them, and evaluated frequently.
H33	S09	SoS capabilities must be constantly analyzed and evaluated.	The capabilities needed to fulfill the SoS missions should be constantly analyzed and monitored by those responsible.
H34	S09	An incremental development and deployment strategy for the SoS should be adopted.	The design process must embrace steps that allow the incremental development of the SoS, promoting and facilitating the dynamism of the architecture.
H35	S09	SoS functions must be revised as the system operates.	SoS dynamics and CS independence can lead to problems in fulfilling the SoS mission over time.
H36	S09	Design decisions must be explicit in the SoS architecture.	During the design, decisions are made to build the SoS architecture. It is necessary to document these decisions.

Continue on next page

Table 4.3 – continued from previous page

ID	Ref.	Description	Rationale
H37	S09	Design decisions must be developed and continually refined.	The independence of CS in SoS can lead to changes in its architecture. New architecture decisions must be re-evaluated and recorded for better SoS maintenance.
H38	S09	SoS scenarios must be developed and continually refined.	Scenarios in which SoS runs are important to verify its effectiveness in the context in which it operates.
H39	S09	A strategy for dynamic integration must be provided.	CS may leave or join SoS at any time. It is necessary to adopt mechanisms to minimize the impact of acquiring or losing capabilities.
H40	S10	Weak emergent behaviors must be identified at design time.	Weak emergent behaviors are those that can be identified at design time. They are not accidental or produce any change in the behavior of the CS. A "strong" behavior can not be predicted at the design phase. It can usually be observed during simulation or in operation.

This SMS aimed to identify heuristics to the design of SoS. The extracted heuristics address SoS initiation, interoperability, characteristics of CS, and governance issues in SoS. We also noted that the studies address mostly directed and acknowledged SoS.

4.3 Focus Group

The focus group allowed us to collect data through interactions in a group of experts and was conducted with planned discussions in order to capture the perceptions regarding the 40 heuristics for SoS design extracted from SMS. Table 4.5 describes the roles used in this focus group.

4.3.1 Planning

We followed the guidelines provided by Beck and Manuel (2008) and Zaganelli et al. (2015). A set of questions was defined to guide the focus group discussions. Figure 4.2 shows focus group phases.

Table 4.4: Which types of heuristics can be applied and how heuristics were evaluated.

ID	Approach	SoS type	Evaluation
S01	Propose SoS-centric middleware services to support the management and execution of SoS in Smart Cities. Further steps proposed to develop case studies in order to validate and evaluate solution.	Collaborative	Not evaluated.
S02	Apply systems thinking and viewing SoS in a sociotechnical context as approach. Heuristics are based on guaranteeing interoperability among CS.	Directed and acknowledged	Case study in the healthcare system and smart home healthcare.
S03	A basic set of architecting principles to assist in SoS design. Refinements of more general heuristics to produce recommendations.	All	Not tested. Formal experiment is not possible due the fact that is not viable duplicate complex systems aiming tests.
S04	Discuss challenges in SoS architecture using SoS-level and CS level, proposing to maintain interfaces in both levels.	All	Not evaluated.
S05	Propose principles to a functional architecture simulator in the context of SoS engineering to define architecture.	Directed and acknowledged	Not evaluated.
S06	Use a sociotechnical ecosystem approach to try to add longevity to the SoS composition process.	Directed	Not evaluated.
S07	Proposed a new model called SApHESIA (SoS Architecting HEuriStIc based on Agents) focusing on environment and its dynamics, using criticality is a metric.	Collaborative and virtual	Simulation using two scenarii to evaluate functional adequation, efficiency, and robustness.
S08	An approach allows for the identification of options capable of mitigating perturbations negatively impacting SoS.	Directed and acknowledged	Case study in maritime security.
S09	A proposal of a "Meta-process for SoS Software Architectures" (SOAR), which supports the authoring of processes to construct SoS software architectures.	Directed and acknowledged	Qualitative survey with experts.
S10	Discuss lack of computational and systems engineering approaches to prevent the engineering of emergent behaviors in SoS modeling and simulation, proposing an approach to deal with this issue.	All	Use of cybernetics, systems, control and network theories.
S11	Presents three architectural styles to realize self-adaptation in SoS. Uses the Same references to the heuristics in S03.	All	Not evaluated.

Phase 1 - Scheduling and invitation

An email was sent with an invitation to participate with suggested dates and times for the meeting.

Table 4.5: Focus group roles.

Role	Description
Moderator	The moderator is responsible for guiding discussions, promoting interaction and the engagement of the participants, enabling the emergence of new ideas about heuristics for SoS design (ZAGANELLI et al., 2015).
Participant	The participant answers the focus group questions and participates in synergy in the discussions that arise. The participants also suggests modifications and improvements in the set of heuristics and the organization of the catalog.

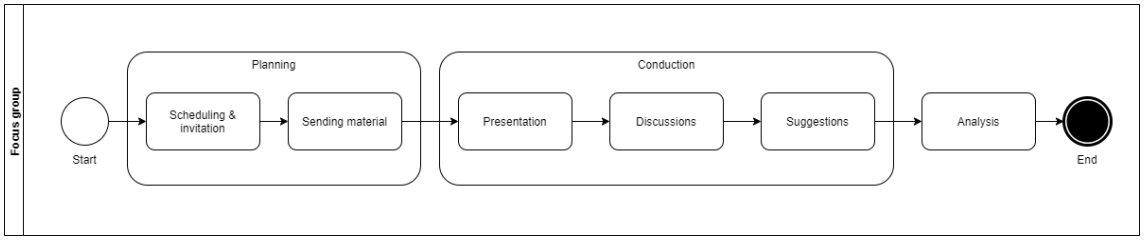


Figure 4.2: Focus group phases.

Phase 2 - Introductory documents

Documents containing the first set of heuristics and the selected studies were sent by email to the participants two days before the focus group to provide preliminary knowledge to support discussions. A Google Meet link was created for the focus group and was also sent by email.

Phase 3 - Presentation

After initial considerations and clarifications about the research and the expected dynamics for the focus group, the document describing the 40 heuristics was presented.

Phase 4 - Discussions

The discussions were prompted by asking participants to answer these six specific questions:

- Q1. Did you understand the formulation and use of the heuristics?
- Q2. Do you think that the heuristics are useful for SoS design?
- Q3. In which types of SoS can this heuristic apply to?
- Q4. Is the quality characteristic indicated for the heuristic consistent?
- Q5. Are there heuristics that can be combined with other heuristics?
- Q6. Are there heuristics that need to be applied in a specific order?

Phase 5 - Suggestions

Based on theoretical knowledge and practical experiences, participants contributed with suggestions regarding heuristics.

Phase 6 - Analysis

After the discussions, the moderator organized the data obtained and conducted a qualitative analysis to clarify how the object of study was perceived by the group (ZAGANELLI et al., 2015).

4.3.2 Execution

We invited 6 people to participate and 4 accepted the invitation. Two participants hold a Master's degree and 2 were Ph.D students. In particular, the focus group aimed to clarify the following points:

1. The importance of the SoS type for the design;
2. In which situations should a heuristic be applied or not;
3. The appropriate sequence of the heuristics;
4. Difference among design, simulation, and execution;
5. Heuristics viewed as means to support the SoS design.

The items are related to interdependencies among heuristics and intended to respond to research question proposed to guide the focus group.

The participants noted that heuristics on interoperability standards specification, standards catalog, and interface identification could be aggregated and placed logically to support SoS design. However, such heuristics should come after the more general ones, such as those that define the SoS goals and the development and maintenance responsibilities.

The ideas raised in the focus group are presented below with a summary of the questions and comments that arose during the discussions among the participants.

The idea of improving explanations on each heuristic was reinforced, aiming to make it easier for the designer to understand and adopt the heuristics. This is important to improve the designer's engagement with the heuristics catalog.

The sequence for the design activities was again reinforced by participants: first, define the type of SoS to be worked on and then concern other issues such as interoperability. Besides this, questions were raised about the separation of heuristics that were specific to SoS from those that were common to traditional systems.

The participants raised questions on the advantages and disadvantages of each CS in participating in the SoS as stated in the H06 heuristic:

- *"Whether the disadvantages of CS participation be mapped, or would it be a concern of each respective stakeholder."*
- *"Could be interesting to combine advantages and disadvantages in a single heuristic."*
- *"In this case, the heuristic could be formulated as identifying which stakeholders participate in the SoS and what are the perceived advantages and disadvantages or which degree of quality is identified in the participation."*
- *"Whether this is a designer's responsibility or just a waste of time in design phase."*

It was observed that heuristic H07 leads to the concept of software costs, which is already a complicated topic for traditional systems, and it is not worth addressing this in SoS design. Also, the perception of stakeholders on how their participation happens when the SoS is executed, and so it is not possible to be done at design time. Open architectures, as described in heuristic H09 seeking to facilitate SoS evolution, can be understood as a choice of SoS engineers, but it could not be considered mandatory. This is a very particular situation for each SoS being designed. The general understanding of the group is that technology issues are not mandatory to be addressed in SoS design.

The heuristic H10, which stated the definition of intermediate SoS forms to facilitate gradual SoS evolution, is considered a significant concern. However, it is impractical at design time because it was often impossible to predict how the SoS evolves to construct this kind of roadmap with intermediate SoS forms.

Heuristic H11, stating the need to know which CS should be helped and which CS should be abandoned, was considered very clear and practical, being related to resilience and maintainability concepts for systems in general.

Defining interfaces already at design time (H12) is related to interoperability and was previously worked in other heuristics statements and respective rationales that mentioned communication standards, catalog of protocols used, data sets, etc. So they should be combined with other heuristics that deal with those issues.

Participants noted that it is very complicated to provide incentive mechanisms for CS' participation in SOS at design time (H13). In addition, this concern is already covered by the heuristics dealing with the benefits of participating in the SoS for the CS and stakeholders.

Participants also noted that it is important to maintain uniformity to the terms treated in the heuristics catalog. Although the content comes from several studies, it is necessary to consistently use terms to avoid confusing readers. It is better to use standardized vocabulary to refer to the respective concept or object. As an example, one study deals with "independent subsystems", which are, in fact, the CS.

It was discussed that heuristics dealing with interoperability are frequently repeated in the heuristics extracted from different studies and should be revised and combined to compose the final heuristics catalog. The same rationale should be applied to the heuristics that deal with testing the CS capabilities.

Heuristics dealing with the identification of changes in CS can be interesting when the designer team has good knowledge of how each CS works, but even so, it is challenging to implement. According to the participants, the heuristic H16 could be modified to deal with changes in CS capabilities, but yet would still be difficult to identify this through an automated method or tool in the design phase.

Validation and verification proposed in the H21 heuristic have to be applied to a well-defined object. Simply saying that there should be a concern for these activities does not help the designer activities. Looking at the traditional software perspective, these are well-defined activities. It would be better to investigate if any issues must be resolved to define how these activities could be conducted in SoS before this heuristic can be applied.

It is necessary to define appropriate metrics before defining and using criticality in SoS, as stated in heuristic H22. Besides this, it is not easy to use this kind of concept in SoS due to its intrinsic evolutionary characteristic.

It is necessary to define what is meant by disturbances in heuristic H23 properly. Treating the disturbances as described in the three-step checklist is a very abstract activity. It was suggested to separate this statement into two heuristics: (i) verifying the points of disturbance and (ii) verifying the architectural alternatives to deal with them. But in this case, this heuristic will be related more to reliability than to interoperability as initially defined.

It was observed that self-regulation of the CS, as stated in the H24 heuristic, shouldn't be a designer's concern. Only heuristics that deal with similar concepts applied to SoS rather than CS can be used in the heuristics catalog.

It was noted that the design, not the solution architecture, should consider a feedback policy for SoS operation. Heuristics 25 to 28 must be removed, as they are directly linked to the intrinsic characteristics of SoS. The heuristic H29 is related to SoS representation and not to SoS design.

The individual capabilities of the CS have to be validated as stated in heuristic H30, but it would be necessary to define validation techniques, which in SoS is a complicated task. It may be useful to check if it is possible to fit this type of activity into a group of activities to create validation policies.

Participants noted that one of the main goals of the SoS project is indeed to identify unwanted emergent behaviors, as stated in H31. It was observed that this heuristic could be adjusted to place this type of behavior in the designer's concerns.

It was noted that the designer would not analyze CS capabilities (H33) due to analyzing and evaluating CS capability should not be done at design time. A better strategy could be to create indicators at design time to make it possible to analyze the SoS at runtime.

Defining an incremental development process, as stated in heuristic H34, is not applicable to the SoS design as it was a broader activity during the whole SoS life cycle, so this heuristic is not applicable to SoS design.

The heuristic H35 does not appear to be valid because there is no reason to review SoS functions as it operates. It should be more useful to know whether there is compliance or adherence of the CS's capabilities to what is expected of them to contribute to SoS fulfilling missions, but that is not what that heuristic says.

Representing design decisions in the architecture, stated in heuristic H36, have

already been addressed in previous heuristics.

Developing and refining design decisions as stated in heuristic H37 is not on the set of the designer tasks and so must be excluded from this heuristics catalog for SoS design.

It is observed that there is no way to handle scenarios in the SoS design mentioned in heuristic H38. It is not possible to point where failures will appear in changing scenarios or how to address those issues at design time. The study S09 (GONÇALVES et al., 2015) does not appear to address the issue in the way this heuristic was presented.

Providing a strategy for dynamic integration proposed in heuristic H39 would be useful, but it couldn't be possible to do this in practice and even more so at design time. This heuristic could be compared to providing mechanisms to promote full interoperability in SoS, which is already one of the grand challenges in SoS.

Listing undesirable emergent behaviors was already discussed in a previous H31 heuristic and can be combined with the H40 heuristic H40 aimed in representing weak emergent behaviors. The text could be rewritten in a more objective way by combining weak and undesirable emergent behaviors in a unique heuristic.

4.3.3 Results and Analysis

After the focus group, it was possible to adjust the set of heuristics, revising the statements and respective rationales, combining some heuristics that deal with similar purposes and removing from the set those heuristics that are not suitable for SoS design.

The focus group discussions also made it possible to create a sequence in which the heuristics should be applied, grouping them into five categories: initiation heuristics (IN), constituent systems heuristics (CS), interoperability heuristics (IO), emerging behaviors heuristics (EB), and monitoring heuristics (MO). Table 4.6 summarize those categories.

With these categories and the adjustments made through the focus group analysis, it was possible to build a refined version of the catalog of heuristics for the design of SoS shown in Table 4.7, as well as to respond to the research question formulated to guide the focus group, defining groups of processes and a logical sequence to apply

Table 4.6: Heuristics categories.

Category	**Description**
Initiation (IN)	Concern activities applicable right at the beginning of the SoS design.
Constituent systems (CS)	Concern activities of the phase of selection of the necessary capabilities for the SoS, as well as which systems can provide them.
Interoperability (IO)	Concern activities that deal with how the CS communicate and what messages and data will be exchanged among them to fulfill the SoS mission.
Emerging behaviors (EB)	Concern activities to describe emerging behaviors that can be identified already at design time, not requiring SoS simulation or execution to be observed.
Monitoring (MO)	Concern activities that shoud be planned to maintain the SoS operation.

the heuristics.

Table 4.7: Refined Heuristics Catalog

ID	**Description**	**Rationale**
IN1	The design should clearly identify who provides the necessary capabilities and resources for the operation of SoS.	SoS depends on the synergy among systems that provide the necessary capabilities for operation, being necessary to guarantee adequate management for this supply.
IN2	The design should clearly consider who is responsible for the construction and operation of SoS.	When funding is required for the SoS operation, stakeholders should be informed how and by whom it will be done so that these issues are dealt with appropriately at the right time.
IN3	The design should clearly identify who benefits from SoS.	Every SoS is built and operates for a purpose. It is important to identify who are the beneficiaries of the activities or operation of SoS.
CS1	Define which capabilities are already available and which should be implemented in the CS for the construction and operation of SoS.	The SoS design should foresee if there are enough functionalities available in the CS to collaborate with the SoS or if it is necessary to implement new functionalities.
CS2	The individual capabilities of each CS should be checked.	Each CS provides to SoS a capability that should be checked if it conforms to what is expected of it.

continue on next page

Table 4.7 – continued from previous page

ID	Description	Rationale
CS3	Design principles that generate the least possible disruption to SoS operation should be applied.	It is necessary to select which design alternative generate less disruption during the SoS operation, defining metrics that make it possible to evaluate the identified alternatives.
IO1	SoS coordination should ensure that each CS can exchange and understand data and messages exchanged among others.	It is necessary that the coordination of the SoS maintains the set of communication standards among CS in order to ensure that each one can communicate with the others when necessary to achieve the purposes of SoS.
IO2	The interfaces of the CS should be defined at design time.	The interfaces of a CS are a crucial factor for the operation of SoS. They are points where the designer can exert influence.
IO3	SoS coordination should ensure that the policies for access and use of the capabilities of each CS can be exchanged and understood by the other CS.	It is necessary to ensure that each CS is able to participate in an SoS, respecting the access and use policies of the other CS. The management and dissemination of these policies is important due to independence characteristics of the CS and the dynamic SoS architecture.
IO4	All data sets to be exchanged among CS should be defined.	The specification of all the data sets to be exchanged among CS should be sufficient to use them to fulfill SoS missions.
EB1	Emergent behaviors should be allocated to the CS requirements.	The emergent behaviors necessary to fulfill SoS missions should be identified and explicitly associated with the respective CS or refinements responsible for them.
EB2	Weak emergent behaviors should be identified at design time.	Weak emergent behaviors are those that can already be identified in the design phase, not requiring SoS simulation or execution to be observed.

continue on next page

Table 4.7 – continued from previous page

ID	Description	Rationale
MO1	SoS missions should be periodically revised as the system evolves.	It is necessary to frequently monitor the fulfillment of the SoS missions since the dynamics of the SoS and the independence of the CS can bring problems to SoS over time.
MO2	The SoS design should include a feedback policy for the operation of the SoS.	It is necessary to monitor the SoS to detect problems during its operation and define the actions required to deal with them.
MO3	The interface patterns that emerged in the evolutionary process should be identified.	The evolutionary process may generate the need to use new communication standards or to update the existing standards. It is necessary to maintain the set of standards used as the evolution of the CS and SoS takes place, generating a roadmap for the process.

In the next section, the evaluation for this heuristic catalog was presented from a survey with SoS experts. The survey had closed questions regarding the heuristic and open questions for general suggestions and comments from the participants.

4.4 Survey

After the focus group, a survey was conducted to evaluate the refined set of heuristics. The survey was based on a catalog of heuristics produced conducting an SMS with further refinement in the focus group dynamics described in the previous section. The survey aimed to evaluate the applicability of the heuristics catalog regarding SoS design from the perspective of experts in the SoS context.

4.4.1 Planning

We chose to invite the researchers that formed the scientific committee of SESoS/WDES 2021[4] as SoS experts to evaluate the heuristics catalog for this evaluation step. The

[4]http://sesos-wdes-2021.icmc.usp.br/Committee.php

researchers' emails were collected from personal websites or the websites of the departments of the respective universities.

The heuristics were grouped into the previously proposed categories of initiation (IN), constituent systems (CS), interoperability (IO), emergent behaviors (EB), and monitoring (MO). For each heuristic, the rationale to apply and an example of a situation where it can be applied were provided.

A data collection form was produced containing the objective of the study, how the set of heuristics was produced, and questions to evaluate each heuristic with a statement and example of use. A pilot was conducted with an SoS expert to correct and adjust the invitation, and the data form before sending an invitation to the population of researchers selected. Appendix I.1 shows the complete data collection form.

The questionnaire was built into three sections. The first contained the free and clear term to participate in the research, while the second section contained questions about the participant's profile. The third section had objective questions for each of the 15 heuristics to be evaluated with the agree, neutral, and disagree options (CHUERI, 2021) and also a field for comments and suggestions about the heuristic being evaluated if necessary. At the end of the questionnaire, an open question was presented for general comments and suggestions.

4.4.2 Execution

In total, we invited 31 researchers, and 15 responded to our questionnaire. Figure 4.3 describes the academic profile of the participants and their familiarity with the SoS theme.

From the graphs presented above, it can be seen that 93.3% of respondents were researchers or professionals with a Ph.D., the majority with good or very good knowledge of SoS (33.3% in both cases), which makes the survey's population very qualified and the responses very representative.

4.4.3 Results

For each heuristic, a question was proposed to know if the participant agreed, was neutral, or disagreed with the heuristic statement and if they had any comments

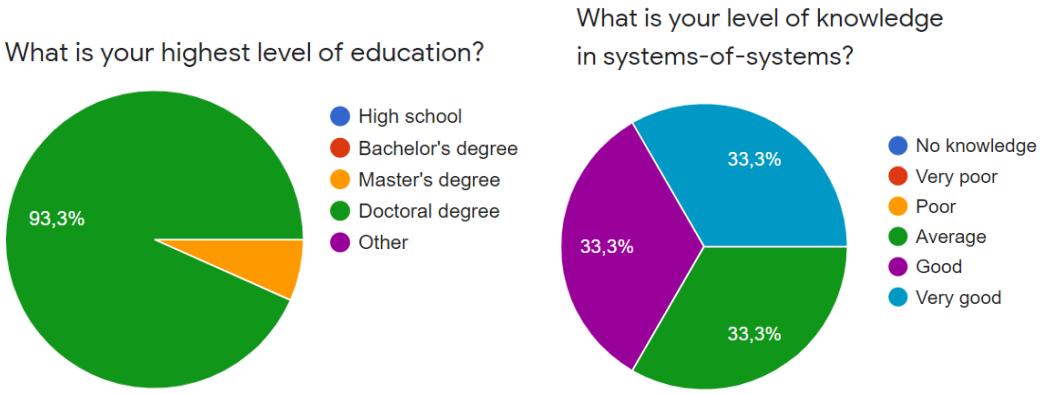


Figure 4.3: Profile of the respondents.

about the statement formulation or suggestions to improve the heuristic statement. Figure 4.4 shows the respective percentages for each answer. The first bar chart represents the responses considering all 15 respondents (group 1), while the bar chart on the right represents only the responses of 10 researchers who declared having good or very good knowledge of SoS (group 2).

The responses and comments were reviewed for each of the catalog's heuristics. The acceptance criterion for the heuristics was greater or equal to 50% of the agreement rate in the two groups. Some heuristics had to be rewritten, considering researchers' comments. Also, according to the researchers, the SoS type of coordination was raised as an important factor when evaluating the applicability of each heuristic.

Heuristic IN1 - The design should clearly identify who provides the necessary capabilities and resources for the operation of the SoS.

Most (80% in group 1 and 70% in group 2) agreed with the heuristic. From the answers and comments, it can be concluded that the heuristic was well mapped, although it is not regarded as valid for virtual SoS, where there is no type of coordination. Also, from the comments, it is noted that the understanding of what a heuristic is can vary depending on the researcher, and it is necessary to describe better the concept of heuristics used in this work.

Another interesting point brought up in the comments drew attention to the fact that this heuristic holds at the design phase, but when SoS begins operation, properties may appear that are not initially foreseen by designers. It may be diffi-

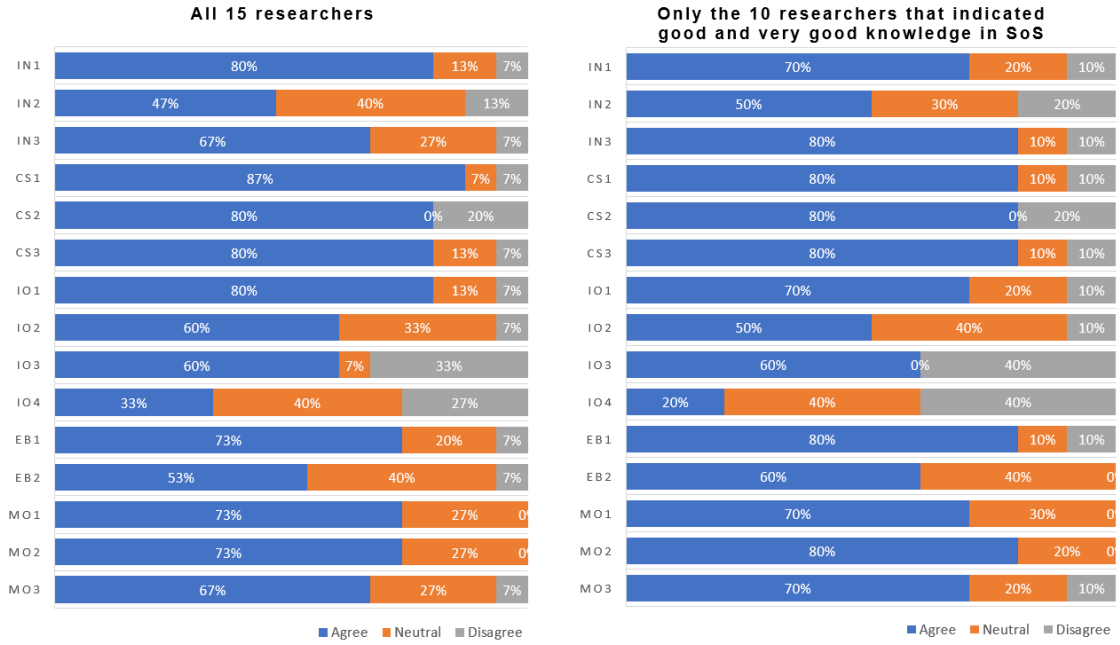


Figure 4.4: Responses to the survey.

cult for this heuristic to remain valid over time while the SoS operates and evolve. Although it is important to know who provides the resources for the functioning of SoS, it is necessary to take into account that there will be unforeseen issues along its evolution cycle since SoS is never fully ready.

IN1 was **accepted**, applied only to directed and acknowledged SoS types.

Heuristic IN2 - The design should clearly consider who is responsible for the construction and operation of the SoS.

Respondents agreed with this heuristic in the following proportion: 47% in group 1 and 50% in group 2. It can be noted from the comments that a highly evident characteristic is the unpredictability in the composition and evolution of the SoS. One of the comments reported the unpredictability of the applications developed and would be part of the Apple store, bringing a parallel with software ecosystems. The idea is that many events occur in a very automatic way in SoS.

A question about the type of SoS coordination was raised again, based on the fact that the different configurations that an SoS can assume can lead to behavior not foreseen by the designers. For example, in a convoy of autonomous vehicles, the decision of which vehicle goes ahead is made based on many factors like fuel consumption, routes, air resistance, and accident risks, resulting in unpredictable

behaviors due to the complexity of all car systems having to work together.

IN2 was **accepted**, applied only to directed, acknowledged, and collaborative SoS types only.

Heuristic IN3 - The design should clearly identify who benefits from SoS.

A total of 67% of group 1 and 80% of group 2 agreed with heuristic IN3. The participants' observations show the dynamic scenario of the SoS, with the change in who benefits from it throughout its evolution, and the application of this heuristic is more appropriate to targeted SoS. The type of SoS coordination is treated again as mandatory for the application of the heuristic.

The comments showed that it might not be possible to identify who benefits from SoS. Furthermore, it is also necessary to understand if the benefit is at the business level, as in the case of a CS, or if it is a social benefit, as in the case of those who are not directly connected to the SoS.

IN3 was **accepted**, applied only to directed and acknowledged SoS types.

Heuristic CS1 - Define which capabilities are already available and which should be implemented in the CS for the construction and operation of SoS.

The major part of researchers agreed with this heuristic (87% in group 1 and 80% in group 2). Despite the high degree of agreement with this heuristic, comments were made about whether this statement is a heuristic or an activity for SoS design, reinforcing the need to define the adopted heuristic concepts better.

Another important question was about the choices of which capabilities exist, and therefore susceptible to reuse, and which will need to be developed. It noted that choosing to reuse available capabilities rather than constructing new ones should be evaluated by designers as these choices can lead to poor performance of the built SoS.

CS1 was **accepted**, applied only to directed, acknowledged, and collaborative SoS types.

Heuristic CS2 - The individual capabilities of each CS should be checked.

For this heuristic, 80% of group 1, and 80% of group 2 agreed with the statement. Despite being well evaluated, this heuristic was criticized in the comments for not being an objective evaluation form, as informed in the definition of heuristic at the beginning of the questionnaire. In addition to the already mentioned need to better

define what heuristic for this work is, it is also necessary to better define what is considered capability, despite the term being well known and applied in SoS.

CS2 was **accepted**, applied only to directed, acknowledged, and collaborative SoS types.

Heuristic CS3 - Design principles that generate the least possible disruption to SoS operation should be applied.

For this heuristic, 80% agreed in group 1, and 80% agreed in group 2. Despite the high degree of agreement, it was suggested to mention which possible topologies an SoS can assume to decide on the most stable one. It was also mentioned that this heuristic is essential for critical systems, which may lead to further classification when considering the application of this heuristic. Once again, the importance of defining the type of coordination existing in the SoS was mentioned.

CS3 was **accepted**, applied only to directed SoS types.

Heuristic IO1 - SoS coordination should ensure that each CS can exchange and understand data and messages exchanged among CS.

The rate of agreement for this heuristic was 80% in group 1 and 70% in group 2. The SoS coordination issue was brought up again in the comments, and it is important to check how to address it in the heuristics catalog. It was suggested to split this heuristic into two parts: one to understand the data and the other to understand the messages, which should make the statement clearer.

IO1 was **accepted**, applied only to directed and acknowledged SoS types.

Heuristic IO2 - The interfaces among the CS should be defined at design time.

The rate of agreement for this heuristic was 60% in group 1 and 50% in group 2. Although half of the participants in group 2 agree with this heuristic, this was one of the most explicit found in the literature. The characterization of the type of SoS coordination was again identified as essential for the application of this heuristic, needing to think about additional layers for connectivity in virtual SoS, not necessary for directed SoS. SoS coordination issues should be addressed in the review of the heuristics catalog.

IO2 was **accepted**, applied only to directed SoS types.

Heuristic IO3 - SoS coordination should ensure that the policies for access and use of the capabilities of each CS can be exchanged and understood by the other

CS.

For this heuristic, 60% agreed in both groups 1 and 2. Besides this, 40% of the group 2 participants disagree with this heuristic, which is a quite negative evaluation. Access policies were recognized as essential to help valuable data protection, but they were pointed out as unimportant for other classes of services like the internet, which the services work perfectly without knowing how each other works. Comments suggest that this heuristic is more adherent to directed SoS, not being possible to apply it to virtual SoS.

IO3 was **accepted**, applied only to directed and acknowledged SoS types.

Heuristic IO4 - All data sets to be exchanged among CS should be defined.

Only 33% of group 1 and 20% of group 2 agreed with this heuristic. According to the comments, applying this heuristic may restrict emergent properties in the SoS. Moreover, CS in virtual SoS can completely ignore being part of the system, making it challenging to define their data sets. Another question was whether data sets or data formats should be defined. IO4 was **not accepted** to be applied to any of the SoS types. For this reason, this heuristic was excluded from the final catalog.

Heuristic EB1 - Emerging behaviors should be allocated to the CS requirements.

Respectively, 73% and 80% agreed with this heuristic in group 1 and group 2. However, it was commented that the term "emergent behavior" was not defined in the initial guidelines, and there was doubt whether the heuristic was saying that the behaviors would be observed or should be indicated in the CS requirements.

EB1 was **accepted**, applied only to directed SoS types.

Heuristic EB2 - Weak emerging behaviors should be identified at design time.

Respectively, 53% and 60% agreed with this heuristic in group 1 and group 2. A comment was made that this heuristic is more easily applied at design time than at the execution time, which, by the way, coincides with the proposal of this catalog. It was also indicated to apply this heuristic in a non-mandatory way.

EB2 was **accepted**, applied only to directed SoS types.

Heuristic MO1 - SoS missions should be periodically revised as the system evolves.

For this heuristic, 73% and 70% agreed in group 1 and group 2. A comment shows an interpretation problem about whether the heuristic is an action that must take place to review the missions or whether it is a behavior to be observed as the

SoS evolves. Other participants mentioned that this heuristic should be a quality criterion for SoS, and it is important to monitor it continuously. However, it was asked how to measure the mission accomplished, what could be used as evidence, and what is the period for these revisions.

MO1 was **accepted**, applied only to directed and acknowledged SoS types.

Heuristic MO2 - The SoS design should include a feedback policy for the operation of the SoS.

Respectively, 73% and 80% agreed with this heuristic in group 1 and group 2. The only comment made was that it could be challenging to provide this type of feedback. This situation can lead to further detailing activities beyond the SoS design, such as actions required when problems are detected.

MO2 was **accepted**, applied only to directed and acknowledged SoS types.

Heuristic MO3 - The interface patterns that emerged in the evolutionary process should be identified.

Group 1 agreed in 67%, and group 2 agreed in 70% with this heuristic. Again the SoS coordination is mentioned as a key factor to make it possible to apply the heuristic. Besides this, a metric was considered an important factor to make this heuristic applicable.

MO3 was **accepted**, applied only to directed, acknowledged and collaborative SoS types.

Table 4.8 summarizes the suitability of each heuristic by type of SoS coordination, taking into account the observations made by the survey participants.

4.5 Limitations

The SMS was an efficient tool for extracting the heuristics but some limitations could be observed. Using the acronym "SoS" in the search string caused the retrieval of studies unrelated to systems-of-systems. Studies with terms like "start of season", "sum of squares", "silicon on solid", "swedish obese subjects" among others, were retrieved which caused unnecessary additional work. On the other hand, other terms like "guidelines" came up during focus group discussions and in survey comments and could be included in the search string, bringing additional studies retrieved in the SMS.

Table 4.8: Classification of heuristics for each type of SoS coordination.

ID	Heuristic	DIR	ACK	COL	VIR
IN1	The design should clearly identify who provides the necessary capabilities and resources for the operation of the SoS.	Valid	Valid	Invalid	Invalid
IN2	The design should clearly consider who is responsible for the construction and operation of the SoS.	Valid	Valid	Valid	Invalid
IN3	The design should clearly identify who benefits from SoS.	Valid	Valid	Invalid	Invalid
CS1	Define which capabilities are already available and which should be implemented in the CS for the construction and operation of SoS.	Valid	Valid	Valid	Invalid
CS2	The individual capabilities of each CS should be checked.	Valid	Valid	Valid	Invalid
CS3	Design principles that generate the least possible disruption to SoS operation should be applied.	Valid	Invalid	Invalid	Invalid
IO1	SoS coordination should ensure that each CS can exchange and understand data and messages exchanged among CS.	Valid	Valid	Invalid	Invalid
IO2	The interfaces among the CS should be defined at design time.	Valid	Invalid	Invalid	Invalid
IO3	SoS coordination should ensure that the policies for access and use of the capabilities of each CS can be exchanged and understood by the others.	Valid	Valid	Invalid	Invalid
IO4	All data sets to be exchanged among CS should be defined.	Invalid	Invalid	Invalid	Invalid
EB1	Emerging behaviors should be allocated to the CS requirements.	Valid	Invalid	Invalid	Invalid
EB2	Weak emerging behaviors should be identified at design time.	Valid	Invalid	Invalid	Invalid
MO1	SoS missions should be periodically revised as the system evolves.	Valid	Valid	Invalid	Invalid
MO2	The SoS design should include a feedback policy for the operation of the SoS.	Valid	Valid	Invalid	Invalid
MO3	The interface patterns that emerged in the evolutionary process should be identified.	Valid	Valid	Valid	Invalid

DIR=directed, ACK=acknowledged, COL=collaborative, and VIR=virtual.

One of the advantages of the focus group is that it offers research participants the opportunity to generate ideas together, creating new insights on a particular subject. Although a small number of participants provides more in-depth discussions on a specific topic, it could be interesting to promote a focus group with more participants to enrich the discussions and increase the ideas generated about using proposed heuristics.

The survey is an extremely useful tool for research. However, particularly in this research, finding practitioners and researchers to respond to surveys was difficult due to the required background knowledge. In addition, there are other barriers, such as respondents considering invitations as unwanted email messages, such as an

invasion of privacy or "junk mail," leading to a low response rate.

4.6 Final Remarks

In this chapter, we extracted, refined, and evaluated the heuristics. The combined procedures of an exploratory study to initially understand SoS concerns, an SMS to find and extract heuristics for SoS from the literature, the organization of the findings through the focus group, and the final evaluation of the heuristics by experts made it possible to build the first version of the heuristics catalog grouped in categories, suggested as a sequence of tasks for the designer.

This SoS heuristics catalog can help designers anticipate problems and concerns at design time, making it possible to mitigate risks before building and operating the SoS. Resources and functionalities can be planned, and building decisions can be made, taking into account a more detailed scenario.

In order to verify how the heuristics can be applied in practice, a tool for SoS modeling was designed and built using the mission model notation of the mKAOS language. The next chapter details the design, construction, and evaluation processes of this tool.

Chapter 5. Modeling Tool

In this chapter, we describe the technological contribution of the research to assist in the design of SoS through the use of part of the constructed heuristics catalog. Section 5.1 brings the introduction of this chapter; in Section 5.2 details the requirements for the tool construction; Section 5.3 shows the tool implementation process; in the Section 5.4 the tool evaluation process is presented; Section 5.5 shows the limitations for this work and Section 5.6 present the final remarks.

5.1 Introduction

The studies described in Chapters 3 and 4 made it possible to construct a heuristic catalog that can be used to help in SoS design. In order to verify how the heuristics can be used in practice, a graphical tool was built that could help users deal with issues that affect SoS even before its construction,

The complete set of heuristics catalog was not implemented in this tool. Some of the heuristics are suitable to be applied to the SoS at design time, being feasible to be used until the beginning of the operational phase of an SoS, but not in the subsequent phases. For example, the heuristic MO1 "SoS missions should be periodically revised as the system evolves", and the heuristic EB1 "Emerging behaviors should be allocated to the CS requirements" could only be applied after an SoS begins operating.

We named this tool mKAOS Studio Lite, inspired by the mKAOS Studio developed by the ConSiste research group at Federal University of Rio Grande do Norte [1]. This chapter describes the requirements defined for this tool, its proposal, how it was implemented, and how it works in practice.

[1] https://www.facebook.com/consiste.ufrn/

5.2 Requirements

The objective of this section is to elicit requirements for the creation of the mKAOS Studio Lite, which is aimed to assist SoS design. The tool should provide a better understanding of how the system works, who is engaged in supplying resources and capabilities, and what the impacts are for SoS design decisions.

Requirement #1 (R1): The tool should use a proper SoS modeling notation.

The use of suitable notation for modeling SoS, which appropriately and clearly represents the interrelationships involved in the SoS, can help in SoS design understanding. Due to the dynamics and characteristics of SoS, we chose to use the mKAOS (SILVA; BATISTA, et al., 2015b) mission model notation that can represent the interrelationships among the CS and the missions that an SoS fulfills.

Requirement #2 (R2): The tool should use widespread technology.

Using widely available technologies can help in the success of the tool dissemination. The mKAOS Studio tool was produced using the Java language [2] and the Eclipse tool [3], both widely used in industry. The Java language, despite its recognized robustness for industrial applications, can present incompatibility problems among its various versions (STEIJGER, 2008), which can make its use complicated in several companies with different technological infrastructures.

The Eclipse tool is one of the most used by the industry specialized in software development, providing not only functionality for managing and editing source code but also providing a vast number of plugins to perform other tasks such as systems modeling software simulation and even network packet analysis. However, Eclipse is not suitable for managers but for developers. Eclipse has high entry barriers such as learning curve, complexity in configuration and use, performance issues, and lack of documentation (KIRTLEY et al., 2008).

Requirement #3 (R3): The tool should use standard interface.

Using available tools and technologies installed by default for most user com-

[2] https://www.java.com/pt-BR/about/
[3] https://www.eclipse.org/

puters can help in the tool use. According to REES (2002), the interface used by modern browsers brings a series of advantages such as:

- Working natively with XML[4] content;
- Ease of use in processing information for communication via firewalls;
- Encapsulating service protocols such as Simple Object Access Protocol (SOAP)[5] and Unified Distributed Data Interchange (UDDI)[6]; and
- Dynamic content based on a Document Object Model (DOM) [7] which allows objects manipulation using script languages like ECMAScript[8].

Requirement #4 (R4): The tool should guide the user through the model-building process.

The tool should follow the evolution of the model construction, guiding the user in making the right connections among elements of the diagram, alerting to inconsistent links such as a CS providing another CS or a capability in the diagram that is not provided by any CS.

Requirement #5 (R5): The tool should use heuristics to evaluate the produced model.

The objective of the catalog of heuristics is to assist in the design phase of SoS projects, serving as rules for an objective form of evaluation, without prior knowledge of SoS goals. The application of the heuristics to verify the model must take into account the order of application of the heuristics, the type of SoS coordination, and the properties of each represented element.

Requirement #6 (R6): The tool should allow remote view of model attributes.

As an additional requirement, to provide model interoperability with other tools. The proposal is that automated mechanisms or remote users can know and interact

[4]https://www.w3.org/TR/xml/
[5]https://www.w3.org/TR/soap/
[6]https://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm
[7]https://dom.spec.whatwg. org/
[8]https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

on-the-fly with the model being designed in order to point out issues to be further addressed, propose improvements or solutions for the architecture, or address issues other than design ones.

5.3 Implementation

The mKAOS Studio Lite tool is based on two modules: a first module that runs on the client that operates the graphical interface for building SoS diagrams and a second module that runs on the server, storing a synchronized version of the model being built on the client and manages a queue of commands that can be sent to the client by a third party with access to an API. Figure 5.1 summarizes messages being sent among server, client and external inspector.

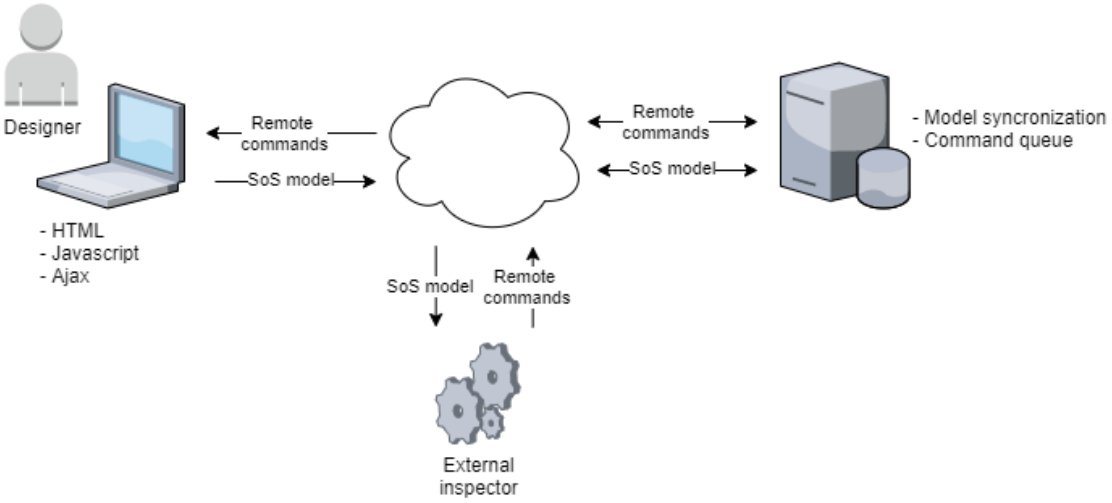


Figure 5.1: mKAOS Studio Lite architecture.

For the server-side tool implementation, an environment with Apache web server, MySQL database, and PHP script language was configured. The database maintains a table with the attributes of diagram elements as the user builds the diagram. Synchronization is performed through a web service called by the client interface every 30 seconds. Also on the server-side, a mechanism was implemented to allow people to send commands remotely to the model, allowing collaboration in the model being drawn in real-time. This remote functionality is implemented through a queue of commands that are processed and sequenced on the server and then sent to the client.

The user interface uses the web browser as a client and is implemented with HTML5[9] language specification, cascade style sheets (CSS), and JavaScript libraries. The tool uses the network module from the vis.js[10] library to implement a network diagram with nodes and edges that are customized in shapes, contours, and colors to represent the elements CS, refinements, missions, and refinement and responsibility links that correspond to the elements used in the mKAOS notation, meeting the requirements R1 and R4.

The tool's interface also uses an event and object model operated with the aid of the jQuery library [11], which also enables asynchronous communication among client and server, using the Ajax protocol (GARRETT et al., 2005). This feature runs natively on all modern browsers without the need to install additional software, which meets R2 and R3 requirements.

The Sweetalert2[12] JavaScript library was used aiming to improve the accessibility of the user interface. The JavaScript libraries used and their respective functionalities were summarized below:

- **jQuery**. It is used to manipulate user interface elements such as forms and buttons as well as to make text input validation;
- **vis.js**. A browser-based dynamic visualization library that utilizes HTML5 canvas element functionality and facilitates manipulation and interaction with dynamic data; and
- **Sweetalert2**. A library that replaces the JavaScript native modal warning and error dialog windows, following the WAI-ARIA[13] framework to improve accessibility in web applications.

The process of applying heuristics to check the model being designed uses the properties of the elements. The model provides visual signaling when it is necessary to provide some data for the elements, so the nodes are only outlined when there is no data provided and are filled with the respective node type color when the mandatory data is provided.

[9]https://html.spec.whatwg.org/multipage/
[10]https://visjs.org/
[11]https://api.jquery.com/jquery.ajax/
[12]https://sweetalert2.github.io/
[13]https://www.w3.org/TR/wai-aria/

The “Check Model” routine was created to check the model, which will be activated at any time from the beginning of the diagram drawing. The “check model” button triggers this routine by doing a complete SoS check at two levels. At the first level, which also meets requirement R4, it checks whether the designed model meets the rules on minimum elements to be observed in an SoS model using mKAOS notation, informing the user what must be done taking into account the current stage of the model. Thus, the suggestions provided are gradually refined as the drawing progresses. Examples of rules used during drawing are:

- There are no SoS without at least one mission;
- There are no SoS without CS; and
- There are no SoS without the refinement of capabilities.

On the second level, the “check model” routine uses heuristics to verify the model and provide feedback to the user. The verification of heuristics uses the assigned properties and relationships among elements to present a report on the issues to be addressed in the design, meeting requirement R5. The heuristics used in the model check are:

- Heuristic IN1 - The design should clearly identify who provides the necessary capabilities and resources for the operation of the SoS.
- Heuristic IN2 - The design should clearly consider who is responsible for the construction and operation of the SoS.
- Heuristic IN3 - The design should clearly identify who benefits from SoS.
- Heuristic CS1 - Define which capabilities are already available and which need to be implemented in the CS for building and operating the SoS.
- Heuristic CS2 - The individual capabilities of each CS should be checked.
- Heuristic IO1 - The interfaces among the CS should be defined at design time.
- Heuristic MO1 - SoS missions should be periodically revised as the system evolves.

- Heuristic MO2 - The SoS design should include a feedback policy for the SoS operation.

Some heuristics from the catalog constructed in Chapter 4 are not used in this tool. The heuristic CS3 was not implemented because it depends on the definition of a set of design principles to be applied to SoS operation, which is not the scope of this study. The heuristics IO2 and IO3 are only applicable from the early phases of SoS construction. The heuristics EB1, EB2, and MO3 are not included in the tool since emergent behaviors and novel interface patterns can only be seen after SoS begins to operate, as a part of the SoS evolutionary process. A summarized snippet with some heuristics is presented in Appendix II.1 for the "check model" routine.

5.4 Evaluation

The evaluation process of the mKAOS Studio Lite tool was carried out through a feasibility study. The tool produced using the catalog of heuristics and the mKAOS notation aimed at helping in Sos design. This graphical tool allows professionals to design a SoS in an interactive process, showing where is a lack of information that should be provided to improve the chances of success of the designed SoS. With the objective of evaluating mKAOS Studio Lite, we conducted a feasibility study using part of the Technology Acceptance Model (TAM) (DAVIS, 1993).

5.4.1 Feasibility study

According to SPINOLA et al. (2008), the use of evidence provided from experimental studies allows the characterization of a technology before its adoption in software projects. We conducted a feasibility study with a series of practical activities in SoS design using the tool. To do this, a series of tasks and a questionnaire were proposed to a group of developers and system engineers.

According to Polanĉiĉ et al. (2010), TAM has the advantage of being focused specifically on information technologies. The questionnaire was produced with part of the TAM (DAVIS, 1993) was applied to the group to verify the viability of the mKAOS Studio Lite. The questions derived from TAM were formulated to evaluate the ease of use, and the usefulness of the tool (SANTOS, 2016).

Planning

A study was planned to evaluate the tool, verifying how professionals in the area of software systems of a company evaluate the usability and usefulness of the tool to model an SoS in a proposed characterization. A script was generated with the basic concepts of SoS and with explanations for understanding the context in which professionals would work in modeling using the tool. The document drawn up for this purpose can be found in Appendix III.1.

In addition, a questionnaire was generated using TAM to address why users accept or reject mKAOS Studio Lite, using two concepts: (i) perception of ease of use; and (ii) perception of usefulness. This questionnaire had four questions related to the ease of use and four questions related to usefulness, both using the Likert scale. Three more open questions are added asking about (i) the positive and negative aspects perceived, (ii) suggestions for improvements to the tool and the set of heuristics, and (iii) general suggestions, difficulties and additional comments about the tool and the study as a whole. The survey model is in Appendix III.3 and the questions are presented in Table 5.1.

A pilot was carried out with a Master's student in order to adjust and correct problems in the research instruments. In this pilot, necessary adjustments in the text were detected, such as the need to incorporate more concepts on SoS and suggestions to modify the evaluation questionnaire, such as transforming the questions into statements in the case of the first 8 questions where there was the application of the Likert scale (Totaly agree, Agree, Neutral, Disagree, and Totaly disagree).

Execution

Fifteen professionals from the institution were invited to participate in the study, of which 7 accepted the invitation. Each participant received instructions by email containing the study proposal, the informed consent term form with the conditions for participation, an introduction to SoS and modeling using the mKAOS notation.

Individual meetings were scheduled to conduct the study with each participant via videoconference. During each meeting of approximately 1 hour, the following activities were carried out:

Table 5.1: Questions derived from TAM model to evaluate the tool.

Question	**Description**	**Dimension**
Q1	It was easy to learn how to use the tool.	Ease of use
Q2	I was able to use the tool the way I wanted.	Ease of use
Q3	I understood what was happening during the interaction with the tool.	Ease of use
Q4	I was able to perform tasks easily using the tool.	Ease of use
Q5	I believe that using the tool with heuristics was useful to represent the SoS in the proposed situation.	Usefulness
Q6	Using the tool allowed us to understand how the CS are related and at which points there may be problems to achieve the global SoS mission.	Usefulness
Q7	The use of heuristics in the tool improved my performance in performing the proposed tasks.	Usefulness
Q8	Using the tool supports IT management activities.	Usefulness
Q9	In your opinion, were positive or negative aspects of using the tool identified? If yes, which one(s)?	(Free text)
Q10	Do you have any suggestions for improving the tool or applying the heuristics? If so, please specify it.	(Free text)
Q11	This space is reserved for any additional comments (difficulties, criticisms and/or suggestions) regarding this study.	(Free text)

- Some clarifications were given on the diversity and complexity of systems in the modern world and how SoS happen in this context. It was explained how the SoS differs from other complex systems and about the managerial and operational independence of the CS. In this phase, the concept of heuristics used in the study was presented, as well as a summary of the process for compiling the set applied to the tool;

- After the clarifications, the tool was presented through an interactive script with a step-by-step on how to model an SoS using an example situation. At each stage, the "Check Model" functionality was used to show how the tool was intended to guide the users in the construction of the SoS model;

- A hands-on session was conducted with user trying the tool, being guided in case of some feature misunderstanding and asking general questions. A tutorial integrated to the system was presented, containing concepts for SoS and heuristics, an example of SoS model, a description of the interface and each of the elements of the tool, a roadmap for building a SoS model and the statement and the rationale for the heuristics evaluated by the tool; and

- Finally, the content, purpose, and confidential nature of the survey were pre-

sented, explaining the purpose of each question and providing the link for completion. The orientation given for the participant to spend around one week testing the tool before responding to the survey is presented in Appendix III.2.

At the end of the tool utilization period, another individual meeting was scheduled to resolve any remaining doubts and provide further clarification for the study if necessary, ensuring that the participant could fill the survey. The tutorial produced for the tool is presented in Appendix II.2.

Results

The participants were characterized as follows according to their professional profile: the major part of participants are in developer positions (71,4%), have more than ten years of experience in information systems (71,4%), and have a Master degree education (57,1%). Figure 5.2 shows all data about the professional profile.

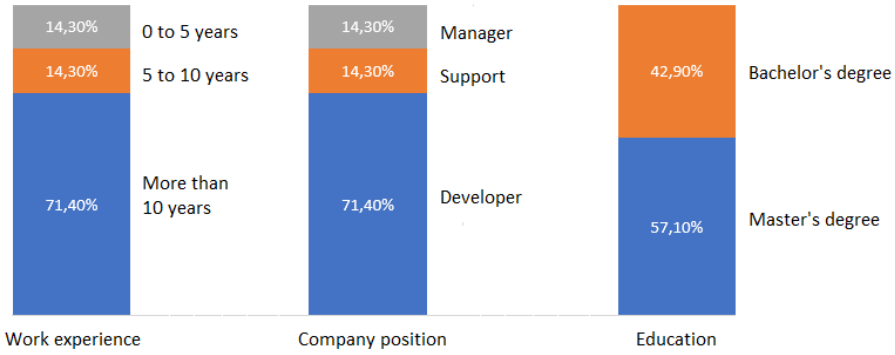


Figure 5.2: Participants experience, position and education.

From the group of participants, 28.6% indicated a good knowledge and 14.3% a very good knowledge in system modeling, totaling only 42.9% of respondents. Only 14.3% of the participants indicated an average knowledge of SoS, and the rest stated poor or non-existent knowledge on the subject.

In the ease of use dimension questions set, only one user (14%) informed that they disagree from Q3, and all others informed they agree or totally agree. In usefulness dimensions, one user (14%) informed that they disagree with the Q6, Q7, and Q8, and all others informed they agree or totally agree. Figure 5.4 summarizes the responses in the 2 dimensions.

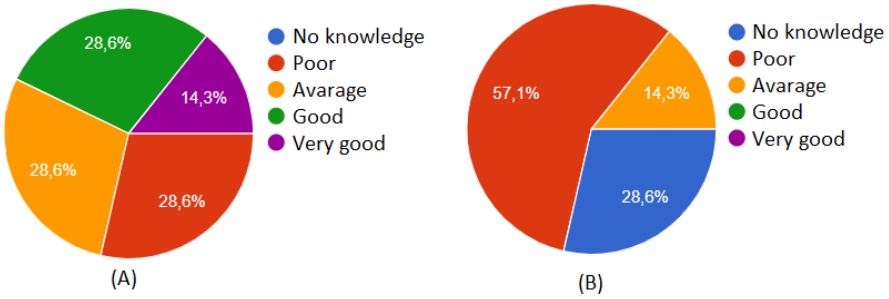


Figure 5.3: (A) Modeling knowledge and (B) SoS knowledge of participants.

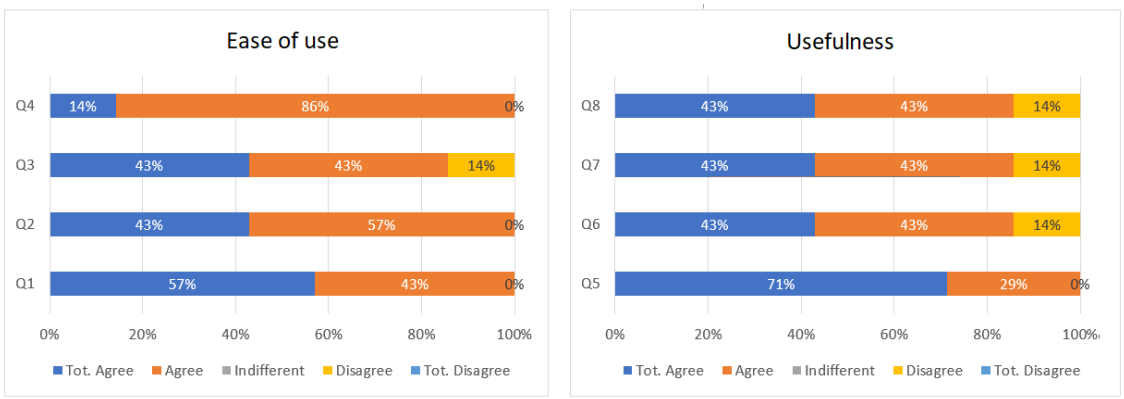


Figure 5.4: TAM model questions.

5.4.2 Defects and improvements

Participants noticed failures in operations and points where it is possible to improve mKAOS Studio Lite. Some of these issues could be corrected and implemented, so the modifications are incorporated in the final version of the tool. Appendix III.4 shows all participants answers. As general positive points in Q9, Q10, and Q11 the participants pointed out the following:

- The tool allows observing weaknesses in the SoS when there is some failure in the CS represented in the model notation;
- The use of symbology with a light background to indicate lack of properties;
- Tool prevents impossible connections between elements; and
- Generated model in JSON format using the QR-CODE link can be useful for further analysis.

As negative points with the possible actions to be implemented are showed in Table 5.2. Some of the errors could be corrected, some suggestions are marked to be evaluated in next versions.

Table 5.2: Questions Q9 Q10 and Q11 errors and suggestions.

Question	**Description**	**Status**
Q9	Delete option is not clear.	DEL key implemented
Q9	Canvas height dimension is too small.	Canvas height increased
Q9	System properties could be mentioned in the tutorial.	Future work
Q9 Q10	Documentation can be improved in the conceptual aspect	Future work
Q9	SoS properties could belong to canvas.	To be evaluated
Q9 Q10	Sometime item placed outside viewport when created.	To be evaluated
Q10	Icons for available and checked missions	To be evaluated
Q10 Q11	Exclusive gateway icon reminds a lot of Cancel or Delete buttons	To be evaluated
Q10	CS interfaces explicit in diagram	To be evaluated
Q10	Undo button	Future work
Q10	Save the model to continue the work	Future work
Q10	Use of AI to apply heuristics	To be evaluated
Q10	Possible interfaces registration	Future work
Q10	Remove debug alerts	Done
Q10	Gateway icon not transparent	Done
Q10	Error and warning heuristics categories	To be evaluated
Q11	Capability without CS not checked	Future work
Q11	Label and Title confusing	Future work

5.5 Limitations

This tool does not implement any integration with other modeling tools, nor does it implement elements of the other five diagrams of mKAOS (SILVA, E., 2015), and it may be necessary to address these issues in the future evolution of this research.

This feasibility study for the tool was conducted in an institution where it was possible to characterize and model an SoS. Looking at the percentages pointed out about knowledge in SoS, we see that the SoS approach of using interoperability between independent systems to build new systems is not a recognized approach among the technicians and managers of the institution, so the concepts and advantages of making such an arrangement are not in the everyday work of the institution's professionals. Also, a little less than half of the participants stated a good and very good knowledge of systems modeling, which undermines the results of this evaluation.

5.6 Final Remarks

This chapter described the implementation of a tool aimed to help SoS designers. The tool was constructed with mKAOS mission model notation and part of the heuristics catalog using a web browser as an interface synchronizing with a backend application running on a server on the internet.

A feasibility study for the built tool was conducted to evaluate the tool and provide insights on how to improve existing functionalities as well as the need to create new functions to serve developers and managers. Despite this, it is noted that it is quite necessary to deepen this assessment with teams and professionals in other institutions where the SoS approach is recognized and used more commonly as a business practice, such as in companies in the military domain.

It would be interesting to conduct other studies, such as those of a participatory nature, in companies with professionals who work with realities that effectively contemplate SoS operations. In this case, the practice could be to follow the work of teams performing design tasks with the help of the modeling tool and heuristics for SoS design, comparing the effect with existing techniques used by the teams without using these artifacts.

In this case, with a more significant number of experienced professionals and teams involved, it would be possible to perform statistical tests to measure the effect of using the tool, verifying whether there is a correlation between the use of the tool to perform design tasks and the expected gain in productivity in design phase of SoS projects.

Chapter 6. Conclusion

This chapter presents the conclusions of this dissertation, the contributions of this research, and the limitations for this study. We also discuss possible future work.

6.1 Epilogue

Information systems are increasingly present in the daily lives of people and companies, performing the most diverse functions such as financial services, health support, improving mobility, and even entertainment. The amount of capabilities provided by these systems makes it increasingly unfeasible to build new systems from scratch, leading designers to think of solutions in the form of system arrangements that integrate the capabilities of several independent systems to deliver new functionality. This is the general idea behind the SoS concept.

But how to guarantee reliability in this type of arrangement? One of the ways to do so is to address issues that affect SoS at design time, which can minimize negative impacts from problems arising in the SoS deployment. For traditional systems, there are already several techniques and tools available to deal with all stages of their life cycle, covering from design to eventual obsolescence. However, for SoS, those tools do not work adequately.

Designing SoS can be a challenge using techniques and tools intended for traditional systems because the degree of uncertainty brought about by the independence of the CS in relation to SoS can be very large.

The main research question in this dissertation (PRQ) was used to guide the investigation of what are the heuristics that can be applied to SoS at design time. To conduct the research, a methodology was planned, conducted, and analyzed involving methods such as exploratory studies, systematic mapping study, focus groups with practitioners, and surveys with experts. The results were an initial set

of 40 heuristics that were organized and refined, producing a catalog of 14 heuristics to be applied to SoS at design time.

The types of SoS to which the heuristics can be applied, dealt with in the secondary research question RQ1, were evidenced during all phases of the research, and it is concluded that the coordination of an SoS must be observed for the application of heuristics according to a sequence of phases in the SoS design. The interdependence between heuristics, dealt with in the secondary research question RQ2, produced this sequence of phases to which heuristics should be applied in the design process: initial definitions, CS characteristics, interoperability issues, identification of emerging behaviors, and concerns about the monitoring of SoS operation.

6.2 Contributions

This dissertation contributed to the construction of a catalog of heuristics that can be applied to verify decisions in the SoS design phase and to identify issues that can be addressed at design time, improving the chances of a successful implementation of SoS, as well as saving time, human and financial resources.

This research also provided to the scientific Information Systems and Software Engineering communities the following detailed contributions:

- A systematic mapping study (Chapter 2, Section 2.3) by which it was possible to extract a set of heuristics for SoS, that was refined by conducting a focus group with Master and Ph.D. students and evaluated in a further survey conducted with SoS experts; and

- A modeling tool for SoS design using the mKAOS notation, running in a web interface. The tool incorporated the SoS heuristics catalog, allowing improve the SoS design process.

6.3 Publications

- **Uma Ferramenta de Modelagem para Análise e Avaliação de Confiabilidade e Interoperabilidade em Sistemas-de-Sistemas por Meio de Heurísticas:** This paper was produced in the definition phase of a research proposal. The article was published at the XIII Workshop on Theses and

Dissertations in Information Systems - WTDSI (IMAMURA; FERREIRA; SANTOS, 2020);

- **Fatores de Governança em Sistemas-de-Sistemas: Análise de uma Instituição Pública Brasileira:** This paper was produced during the first year of research in the initial phase of the research to explore how professionals perceive governance in SoS. The study was published at the V Workshop on Social, Human and Economic Aspects of Software - WASHES (IMAMURA; COSTA, et al., 2020);

- **System-of-Systems Reliability: An Exploratory Study in a Brazilian Public Organization:** This paper was produced in the second year of the research when it was necessary to explore how modeling could help in the characterization of SoS problems. The study was published in the XVII Brazilian Symposium on Information Systems - SBSI (IMAMURA; FERREIRA; FERNANDES, et al., 2021).

6.4 Limitations

The restrictions imposed by the COVID-19 pandemic hampered studies with a larger number of participants and with face-to-face interactions' modality. A participative case study could be conducted, for example, to evaluate the tool produced, following more closely two or more teams of experts within companies that actually work with the SoS approach. However, even with the end of sanitary restrictions, it can be difficult to find companies that use the SoS approach in a structured way and that the administrators agree to participate in studies together with academia.

6.5 Future Work

The heuristics identified by this research are related to SoS design. It may be interesting to study if other types of heuristics are useful for working with SoS in other phases, such as when strategies are needed to monitor and control SoS missions or how to define indicators for the fulfillment of SoS missions.

The tool produced within the scope of this research can be expanded to work in conjunction with other tools in other areas, such as code generation for automating

tests and SoS simulation. It can be useful to work on integrating this tool with a more robust platform such as mKAOS Studio itself and Eclipse framework.

There are some research centers that deal with various questions about SoS, including in different domains such as the military domain. Working together with other groups can produce a synergy capable of leveraging new research on the SoS area.

SoS is still little explored both in industry and in academia, which increases the need for further research on this topic. This research investigated how to improve professionals' understanding of heuristics for SoS design in order to improve the chances of success for SoS implementation.

References

ABBOTT, Russ. Open at the top; open at the bottom; and continually (but slowly) evolving. In: IEEE. PROCEEDINGS of the International Conference on System of Systems Engineering (IEEE/SMC). Los Angeles, USA: [s.n.], 2006. 6–pp.

ACKOFF, Russell L. Towards a system of systems concepts. **Management science**, INFORMS, v. 17, n. 11, p. 661–671, 1971.

BAR-YAM, Yaneer. The Characteristics and Emerging Behaviors of System of Systems. **NECSI: Complex Physical, Biological and Social Systems Project (January 7)**, 2004.

BASILI, Victor R; CALDIERA, Gianluigi; ROMBACH, H Dieter. The goal question metric approach. **Encyclopedia of software engineering**, p. 528–532, 1994.

BATISTA, Thais. Challenges for SoS architecture description. In: PROCEEDINGS of the First International Workshop on Software Engineering for Systems-of-Systems. Montpellier, France: [s.n.], 2013. p. 35–37.

BECK, Susan E; MANUEL, Kate. **Practical research methods for librarians and information professionals**. [S.l.]: Neal-Schuman Publishers New York, 2008.

BOARDMAN, J; SAUSER, B. System of Systems - the meaning of of. In: PROCEEDINGS of the International Conference on System of Systems Engineering (SoSE). Los Angeles, USA: [s.n.], 2006. p. 118–123.

BOSCARIOLI, Clodis; ARAUJO, Renata Mendes; MACIEL, Rita Suzana Pitangueira. I GranDSI-BR Grand Research Challenges in Information Systems in Brazil 2016-2026. **Special**

Committee on Information Systems (CE-SI). Brazilian Computer Society (SBC), p. 12–40, 2017.

BOULDING, Kenneth E. General systems theory—the skeleton of science. **Management science**, INFORMS, v. 2, n. 3, p. 197–208, 1956.

BOUZIAT, Teddy; CAMPS, Valérie; COMBETTES, Stéphanie. A cooperative SoS architecting approach based on adaptive multi-agent systems. In: IEEE. PROCEEDINGS of the International Workshop on Software Engineering for Systems-of-Systems (SESoS). Gothenburg, Sweden: [s.n.], 2018. p. 8–16.

CADAVID, Héctor; ANDRIKOPOULOS, Vasilios; AVGERIOU, Paris. Architecting systems of systems: A tertiary study. **Information and Software Technology**, v. 118, p. 106202, 2020.

CARLOCK, Paul G; FENTON, Robert E. System of Systems (SoS) enterprise systems engineering for information-intensive organizations. **Systems engineering**, Wiley Online Library, v. 4, n. 4, p. 242–261, 2001.

CHUERI, Luciana. **SIDE: A Framework for managing social innovation digital ecosystems**. 2021. 208 pp. PhD thesis – Universidade Federal do Estado do Rio de Janeiro.

CHURCHER, Neville; FRATER, Sarah; HUYNH, Cong Phuoc; IRWIN, Warwick. Supporting OO design heuristics. In: IEEE. PROCEDDINGS of the Australian Software Engineering Conference (ASWEC). Melbourne, Australia: [s.n.], 2007. p. 101–110.

CLARK, John O. System of systems engineering and family of systems engineering from a standards, V-model, and dual-V model perspective. In: IEEE. PROCEEDINGS of the Annual IEEE Systems Conference. Vancouver, Canada: [s.n.], 2009. p. 381–387.

DAHMANN, Judith S.; BALDWIN, Kristen J. Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering. In: PROCEEDINGS of the IEEE Systems Conference. Montreal, Canada: [s.n.], 2008. p. 1–7.

DASTON, Lorraine J. Probabilistic expectation and rationality in classical probability theory. **Historia mathematica**, Elsevier, v. 7, n. 3, p. 234–260, 1980.

DAVIS, Fred D. User acceptance of information technology: system characteristics, user perceptions and behavioral impacts. **International journal of man-machine studies**, Elsevier, v. 38, n. 3, p. 475–487, 1993.

GARCÉS, Lina; NAKAGAWA, Elisa Yumi. A process to establish, model and validate missions of systems-of-systems in reference architectures. In: PROCEEDINGS of the Symposium on Applied Computing (SAC). Marrakech, Morocco: [s.n.], 2017. p. 1765–1772.

GARRETT, Jesse James et al. Ajax: A new approach to web applications. San Francisco, CA, USA, 2005.

GIGERENZER, Gerd. Why heuristics work. **Perspectives on psychological science**, SAGE Publications Sage CA: Los Angeles, CA, v. 3, n. 1, p. 20–29, 2008.

GIGERENZER, Gerd; GAISSMAIER, Wolfgang. Heuristic decision making. **Annual review of psychology**, Annual Reviews, v. 62, p. 451–482, 2011.

GONÇALVES, Marcelo Benites; OQUENDO, Flavio; NAKAGAWA, Elisa Yumi. A meta-process to construct software architectures for system of systems. In: PROCEEDINGS of the ACM Symposium on Applied Computing (SAC). Salamanca, Spain: [s.n.], 2015. p. 1411–1416.

IMAMURA, Marcio; COSTA, Luiz Alexandre; PEREIRA, Bruno; FERREIRA, Francisco Henrique; FONTAO, Awdren; SANTOS, Rodrigo. Fatores de Governança em Sistemas-de-Sistemas: Análise de uma Instituição Pública Brasileira. In: SBC. PROCEEDINGS of the Anais do Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES). Cuiabá, Brazil: [s.n.], 2020. p. 31–40.

IMAMURA, Marcio; FERREIRA, Francisco Cerdeira; SANTOS, Rodrigo Pereira dos. Uma Ferramenta de Modelagem para Analise e Avaliação de Confiabilidade e Interoperabilidade em Sistemas-de-Sistemas por Meio de Heurısticas. In: SBC. PROCEEDINGS of

the Workshop de Teses e Dissertações em Sistemas de Informação (WTDSI) e Anais Estendidos do Simpósio Brasileiro de Sistemas de Informação (SBSI). São Bernardo do Campo, Brazil: [s.n.], 2020. p. 48–51.

IMAMURA, Marcio; FERREIRA, Francisco Henrique; FERNANDES, Juliana Costa; SANTOS, Rodrigo. System-of-Systems Reliability: An Exploratory Study in a Brazilian Public Organization. In: PROCEEDINGS of the Brazilian Symposium of Information Systems (SBSI). Uberlândia, Brazil: [s.n.], 2021.

JACKSON, Scott; FERRIS, Timothy L. J. Resilience principles for engineered systems. **Systems Engineering**, v. 16, n. 2, p. 152–164, 2013. ISSN 10981241.

JACOB, François. The Logic of Living Systems: A History of Heredity. Allen Lane, 1974.

JAMSHIDI, Mo. System of systems engineering-New challenges for the 21st century. **IEEE Aerospace and Electronic Systems Magazine**, IEEE, v. 23, n. 5, p. 4–19, 2008.

KAZMAN, Rick; SCHMID, Klaus; NIELSEN, Claus Ballegaard; KLEIN, John. Understanding patterns for system of systems integration. In: IEEE. PROCEEDINGS of the International Conference on System of Systems Engineering (SoSE). Hawaii, USA: [s.n.], 2013. p. 141–146.

KIRTLEY, Nick; KAMAL, Ahmad Waqas; AVGERIOU, Paris. **Developing a modeling tool using eclipse**. [S.l.]: University of Groningen, Johann Bernoulli Institute for Mathematics and ..., 2008.

KITZINGER, Jenny. The methodology of focus groups: the importance of interaction between research participants. **Sociology of health & illness**, Wiley Online Library, v. 16, n. 1, p. 103–121, 1994.

LIANG, Qianhui; RUBIN, Stuart H. Randomisation in designing software tests for systems of systems. **International Journal of Information and Decision Sciences**, Inderscience Publishers Ltd, v. 4, n. 2-3, p. 108–129, 2012.

LOPES, Frederico; LOSS, Stefano; MENDES, Altair; BATISTA, Thais; LEA, Rodger. SoS-centric middleware services for interoperability in smart cities systems. In: PROCEEDINGS of the International Middleware Conference. Trento, Italy: [s.n.], 2016. p. 1–6.

MADNI, Azad M; SIEVERS, Michael. System of systems integration: Key considerations and challenges. **Systems Engineering**, Wiley Online Library, v. 17, n. 3, p. 330–347, 2014.

MAHMOOD, Asif. 'SoS call'at the other edge of chaos. **Journal of Systems Science and Complexity**, Springer, v. 29, n. 1, p. 133–150, 2016.

MAIER, Mark W. Architecting Principles for Systems-of-Systems. In: WILEY ONLINE LIBRARY, 1. PROCEEDINGS of the International Symposium (INCOSE). [S.l.: s.n.], 1996. v. 6, p. 565–573.

_____._____. **Systems Engineering**, Wiley Online Library, v. 1, n. 4, p. 267–284, 4 1998.

MANTHORPE, William HJ. The emerging joint system of systems: A systems engineering challenge and opportunity for APL. In: 3. PROCEEDINGS of the Johns Hopkins APL Technical Digest. Maryland, USA: [s.n.], 1996. v. 17, p. 305–313.

MCDERMOTT, Tom. Developing systems thinking skills using healthcare as a case study. In: IEEE. PROCEEDINGS of the Conference on System of Systems Engineering (SoSE). Paris, France: [s.n.], 2018. p. 240–244.

MITTAL, Saurabh; RAINEY, Larry. Harnessing emergence: The control and design of emergent behavior in system of systems engineering. In: PROCEEDINGS of the Conference on Summer Computer Simulation (SummerSim). San Diego, United States: [s.n.], 2015. p. 1–10.

NCUBE, Cornelius; LIM, Soo Ling. On systems of systems engineering: A Requirements engineering perspective and research agenda. In: IEEE. PROCEEDINGS of the International Requirements Engineering Conference (RE). Banff, Canada: [s.n.], 2018. p. 112–123.

NIELSEN, Claus Ballegaard; LARSEN, Peter Gorm; FITZGERALD, John; WOODCOCK, Jim; PELESKA, Jan. Systems of systems engineering: basic concepts, model-based techniques, and research directions. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 48, n. 2, p. 1–41, 2015.

NIELSEN, Jakob. **10 Usability Heuristics for User Interface Design**. [S.l.: s.n.], 2005. Available from: ¡https://www.nngroup.com/articles/ten-usability-heuristics/¿.

_____. Enhancing the explanatory power of usability heuristics. In: PROCEEDINGS of the Conference on Human Factors in Computing Systems (SIGCHI). Boston, USA: [s.n.], 1994. p. 152–158.

NULTY, Duncan D. The adequacy of response rates to online and paper surveys: what can be done? **Assessment & evaluation in higher education**, Routledge, v. 33, n. 3, p. 301–314, 2008.

O'BRIEN, James A; MARAKAS, George M. **Management information systems**. [S.l.]: McGraw-Hill/Irwin, 2011. v. 9.

OMG; PARIDA, R; MAHAPATRA, S. Business process model and notation (BPMN) version 2.0. **Object Management Group**, 2011.

POLANČIČ, Gregor; HERIČKO, Marjan; ROZMAN, Ivan. An empirical examination of application frameworks success based on technology acceptance model. **Journal of systems and software**, Elsevier, v. 83, n. 4, p. 574–584, 2010.

REES, Michael J. Evolving the browser towards a standard user interface architecture. In: CITESEER. PROCEEDINGS of the Australasian Conference on User Interfaces (AUIC). Melbourne, Australia: [s.n.], 2002. v. 20, p. 1–7.

RICCI, Nicola; ROSS, Adam M; RHODES, Donna H. A generalized options-based approach to mitigate perturbations in a maritime security system-of-systems. **Procedia Computer Science**, Elsevier, v. 16, p. 718–727, 2013.

RIEL, Arthur. **Object-Oriented Design Heuristics**. [S.l.]: Addison-Wesley, 1996.

SAGE, Andrew P; CUPPAN, Christopher D. On the systems engineering and management of systems of systems and federations of systems. **Information knowledge systems management**, IOS Press, v. 2, n. 4, p. 325–345, 2001.

SANTOS, Rodrigo Pereira dos. **Managing and monitoring software ecosystem to support demand and solution analysis**. 2016. 228 pp. PhD thesis – Universidade Federal do Rio de Janeiro.

SCHNEIDER, Jean-Philippe; TEODOROV, Ciprian; SENN, Eric; CHAMPEAU, Joël. Towards a dynamic infrastructure for playing with systems of systems. In: PROCEEDINGS of the European Conference on Software Architecture. Vienna, Austria: [s.n.], 2014. p. 1–4.

SHARAWI, Abeer; SALA-DIAKANDA, Serge N; DALTON, Adam; QUIJADA, Sergio; YOUSEF, Nabeel; RABELO, Luis; SEPULVEDA, Jose. A distributed simulation approach for modeling and analyzing systems of systems. In: IEEE. PROCEEDINGS of the Winter Simulation Conference (WSC). Monterey, USA: [s.n.], 2006. p. 1028–1035.

SHENHAR, Aaron. A new systems engineering taxonomy. In: PROCEEDINGS of the International Symposium of the National Council on System Engineering (INCOSE). San Jose, USA: [s.n.], 1994. v. 2, p. 261–276.

SHERMAN, Steven J; CORTY, Eric. Cognitive heuristics. Lawrence Erlbaum Associates Publishers, 1984.

SILVA. **Mission-driven software-intensive system-of-systems architecture design**. 2018. 212 pp. PhD thesis – Université de Bretagne Sud; Universidade federal do Rio Grande do Norte ...

SILVA, Eduardo. **Uma Linnguagem para Descrição de Missões em Sistemas-de-Sistemas**. 2015. Master's dissertation – Federal University of Rio Grande do Norte, Rio Grande do Norte, Brazil.

SILVA, Eduardo; BATISTA, Thais; OQUENDO, Flavio. A mission-oriented approach for designing system-of-systems. In: IEEE. PROCEEDINGS of the System of Systems Engineering Conference (SoSE). Västerås, Sweden: [s.n.], 2015. p. 346–351.

_____._____. In: PROCEEDINGS of the System of Systems Engineering Conference (SoSE). San Antonio, USA: [s.n.], 2015. p. 346–351.

SILVA, Eduardo; CAVALCANTE, Everton; BATISTA, Thais; OQUENDO, Flavio; DELICATO, Flavia C; PIRES, Paulo F. On the characterization of missions of systems-of-systems. In: PROCEEDINGS of the European Conference on Software Architecture Workshops. Vienna, Austria: [s.n.], 2014. p. 1–8.

SILVA AMORIM, Simone da; ALMEIDA, Eduardo Santana de; MCGREGOR, John D; FLACH G. CHAVEZ, Christina von. When ecosystems collide: making systems of systems work. In: PROCEEDINGS of the European Conference on Software Architecture (ECSA). Vienna, Austria: [s.n.], 2014. p. 1–4.

SPINOLA, Rodrigo O; DIAS-NETO, Arilo C; TRAVASSOS, Guilherme H. Abordagem para desenvolver tecnologia de software com apoio de estudos secundários e primários. In: SN. PROCEEDINGS of Experimental Software Engineering Latin American Workshop (ESELAW). São Carlos, Brazil: [s.n.], 2008. p. 25.

STEIJGER, Tamara. **Downgrading Java 5.0 Projects: An approach based on source-code transformations**. [S.l.: s.n.], 2008.

SUNSTEIN, Cass R. Moral heuristics. **Behavioral and brain sciences**, [New York]: Cambridge University Press, 1978-, v. 28, n. 4, p. 531–541, 2005.

TVERSKY, Amos; KAHNEMAN, Daniel. Judgment under uncertainty: Heuristics and biases. **science**, American association for the advancement of science, v. 185, n. 4157, p. 1124–1131, 1974.

VAN LAMSWEERDE, Axel. Goal-oriented requirements engineering: A guided tour. In: IEEE. PROCEEDINGS of the IEEE International Symposium on Requirements Engineering. Toronto, Canada: [s.n.], 2001. p. 249–262.

WEYNS, Danny; ANDERSSON, Jesper. On the challenges of self-adaptation in systems of systems. In: PROCEEDINGS of the International Workshop on Software Engineering for Systems-of-Systems (SESoS. Montpellier, France: [s.n.], 2013. p. 47–51.

ZAGANELLI, Bárbara Martins; NISENBAUM, Moises Andre; ALVES, Karla dos Santos Guterres; MARQUES, Sarah Barreto; OLINTO, Gilda. O grupo focal na Ciência da Informação. **Informação & Sociedade**, Universidade Federal da Paraıba-Programa de Pós-Graduação em Ciência da ..., v. 25, n. 3, 2015.

Appendices

Appendix I. Survey to evaluate heuristics

I.1 Data collection form

The 11 pages of the data collection form for the heuristic evaluation survey are presented in the next pages with dummy marks on the questions.

Heuristics for Systems-of-Systems Projects

This survey is part of a master's research project established as a joint effort between the Federal University of State of Rio de Janeiro and Federal Institute of Piauí. The research is conducted by the master student Marcio Imamura under the supervision of Professor Rodrigo Santos and in collaboration with the PhD candidate Francisco Henrique Ferreira and the master Juliana Fernandes. The objective of this research is to collect the opinions of experts on heuristics for Systems-of-Systems (SoS).

Personal identification will not be revealed in the survey report, which will preserve the respondents' anonymity and confidentiality. Your contribution is extremely important to this research, but you can interrupt your participation at any moment. Your participation in the survey will take about fifteen minutes.

Thank you in advance for your kind collaboration!

Marcio Imamura (UNIRIO)
Juliana Fernandes (IFPI)
Francisco Henrique Ferreira (UNIRIO)
Rodrigo Santos (UNIRIO)

SECTION 1. Term of Free and Clarified Agreement (TFCA)

By answering this questionnaire, you allow researchers to obtain, use and disclose the information generated from the data grouped as described below.

CONDITIONS

1. I understand that all information is confidential. I will not be personally identified and agree to complete the questionnaire for research purposes. Data derived from this survey can be published in journals, conferences, and blog posts.

2. I understand that my participation in this survey is entirely voluntary and that refusing to participate will not involve a penalty or loss of benefits. If I choose, I can withdraw my entry at any time. I also understand that if I decide to participate, I can refuse to answer open-ended questions that I don't feel comfortable with.

3. I understand that I can contact the researcher if I have any questions about the research. I am aware that my consent will not directly benefit me. I am also aware that the author will keep the data in a grouped way, collected in perpetuity, and will be able to use it for future academic works.

4. As I move on to the next section, I freely acknowledge my rights as a voluntary research participant, as described above, and provide consent to the researcher to use my data in conducting studies on the area mentioned above.

The questionnaire will be presented after the acceptance of this TFCA. *

- [x] I accept the conditions described in the TFCA.

SECTION 2. Academic Profile

Email

(If you wish to receive the survey report in the future)

What is your highest level of education? *

- ○ High school
- ○ Bachelor's degree
- ○ Master's degree
- ○ Doctoral degree
- ◉ Other

If you work in any sector at the industry, please indicate below.

What is your level of knowledge in systems-of-systems? *

- ○ No knowledge
- ○ Very poor
- ○ Poor
- ○ Average
- ○ Good
- ◉ Very good

SECTION 3. Evaluation of the heuristics

The concepts presented below are fundamental for a better understanding of the research:

SYSTEMS-OF-SYSTEMS
A system-of-systems (SoS) can be defined as an arrangement of independent systems that work in synergy to fulfill a mission that cannot be accomplished by any of the systems in isolation. SoS can be observed in several domains such as urban mobility, health, and smart cities.

HEURISTICS
Heuristics are efficient cognitive processes, conscious or unconscious, that ignore part of the information and can be used in several areas of knowledge as a tool to provide:

- A good approximation instead of the perfect solution to a problem.
- A "well-calibrated guess" for decision-making.
- Objective form of evaluation with no need to know the problem in depth.

The 15 SoS heuristics presented below were grouped into initialization (IN), constituent systems (CS), interoperability (IO), emergent behavior (EB), and monitoring (MO).

Initialization heuristics (IN)

Concern activities applicable right at the beginning of the SoS project.

Heuristic IN1 - The project should clearly identify who provides the necessary capabilities and resources for the operation of the SoS. *

SoS depends on the synergy among systems that provide the necessary capabilities for operation, being necessary to guarantee adequate management for this supply. Example: in an SoS designed for treatment and prevention of natural disasters, where to get the rainfall level of a given region with the necessary update frequency?

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

_____

Heuristic IN2 - The project should clearly consider who is responsible for the construction and operation of the SoS. *

When funding is required for the SoS operation, stakeholders should be informed how and by whom it will be done so that these issues are dealt with appropriately at the right time. Example: how much do they cost, and who will pay for the activities and resources needed to build and maintain the SoS, in addition to those already performed by the constituent systems (CS)?

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

Heuristic IN3 - The project should clearly identify who benefits from SoS. *

Every SoS is built and operates for a purpose. It is important to identify who are the beneficiaries of the activities or operation of the SoS. Example: who are the possible users of SoS, and what is the value of what SoS produces to them?

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

Heuristics on constituent systems (CS)

Concern activities of the phase of selection of the necessary capabilities for the SoS, as well as which systems can provide them.

Heuristic CS1 - Define which capabilities are already available and which should be implemented in the CS for the construction and operation of SoS. *

The SoS project should foresee if there are enough functionalities available in the CS to collaborate with the SoS or if it is necessary to implement new functionalities. For example, in the refinements of 2 or more CS capabilities to produce new information or behavior, additional resources and processing may be necessary.

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

Heuristic CS2 - The individual capabilities of each CS should be checked. *

Each CS provides to SoS a capability that should be checked if it conforms to what is expected of it. For example, the designer should check that the capability provided by a location system is available as often as necessary and is accurate enough for its purpose within the SoS.

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

Heuristic CS3 - Design principles that generate the least possible disruption to SoS operation should be applied. *

It is necessary to select which design alternative generates less disruption during the SoS operation, defining metrics that make it possible to evaluate the identified alternatives. For example: for a critical mission that involves saving lives, it is necessary to apply metrics that involve risk x benefit for the selection of CS and possible redundancy of capabilities.

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

_____

Interoperability heuristics (IO)

Concern activities that deal with how the CS communicates and what messages and data will be exchanged between them to fulfill the SoS mission.

Heuristic IO1 - SoS coordination should ensure that each CS can exchange and understand data and messages exchanged between constituents. *

It is necessary that the coordination of the SoS maintains the set of communication standards between CS in order to ensure that each constituent can communicate with the others when necessary to achieve the purposes of the SoS. For example, an automatic train control system should clearly understand what to do when triggered by other systems to prevent accidents.

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

Heuristic IO2 - The interfaces between the constituent systems should be defined at design time. *

The interfaces between the CS are a crucial factor for the operation of the SoS. They are points where the designer can exert influence. For example, the necessary communication between 2 CS is already available in a satisfactory way to the SoS operation, or is it necessary to design additional connectivity and new communication channels?

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

Heuristic IO3 - SoS coordination should ensure that the policies for access and use of the capabilities of each CS can be exchanged and understood by the other CS. *

It is necessary to ensure that each CS is able to participate in the SoS, respecting the access and use policies of the other CS. The management and dissemination of these policies is important due to the SC's independence characteristics and the dynamic SoS architecture. Example: is it necessary to define any security or traceability mechanism to consume information from a specific CS?

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

Heuristic IO4 - All data sets to be exchanged between CS should be defined. *

The specification of all the data sets to be exchanged between CS should be sufficient to use them to fulfill the SoS missions. Example: the data set provided by CS 1 is unequivocally understood by CS 2 when necessary for the correct SoS operation?

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

---

Heuristics on emerging behaviors (EB)

Concern activities to describe emerging behaviors that can be identified already at design time, not requiring SoS simulation or execution to be observed.

Heuristics EB1 - Emerging behaviors should be allocated to the CS requirements. *

The emerging behaviors necessary to fulfill the SoS missions should be identified and explicitly associated with the respective CS responsible for them. For example, within an SoS that takes care of emergencies in smart cities, which CS selects professionals, equipment, and other resources necessary to respond to an incident?

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

Heuristic EB2 - Weak emerging behaviors should be identified at design time. *

Weak emerging behaviors are those that can already be identified in the design phase, not requiring SoS simulation or execution to be observed. Example: is it possible to determine when there is an advantage for a CS to participate in the SoS, consuming capabilities generated by the SoS for its own purposes?

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

---

Monitoring heuristics (MO)

Concern activities that shoud be planned to maintain the SoS operation.

Heuristic MO1 - SoS missions should be periodically revised as the system evolves. *

It is necessary to frequently monitor the fulfillment of the SoS missions since the dynamics of the SoS and the independence of the CS can bring problems to the SoS over time. For example: in an SoS that helps prevent flooding, is it necessary to add a region to be monitored where a new dam is being built?

- (●) Agree
- ( ) Neutral
- ( ) DIsagree

If necessary, justify your answer and/or suggest modifications

Heuristic MO2 - The SoS project should include a feedback policy for the operation of the SoS. *

It is necessary to monitor the SoS to detect problems during its operation and define the actions required to deal with them. For example: is it necessary to monitor the average time of emergency medical care and the respective vacation schedule of professionals to adjust public health activities in a smart city?

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

Heuristic MO3 - The interface patterns that emerged in the evolutionary process should be identified. *

The evolutionary process may generate the need to use new communication standards or to update the existing standards. It is necessary to maintain the set of standards used as the evolution of the CS and SoS takes place, generating a roadmap for the process. For example: when a new hospital becomes part of the municipality's health system, it may be necessary to include new communication standards in the current health system.

- (●) Agree
- ( ) Neutral
- ( ) Disagree

If necessary, justify your answer and/or suggest modifications

Comments and suggestions

Please report any additional comments and/or suggestions for this survey

Thank you

Thank you for your participation. The results of this survey will be used in conjunction with other research tools to produce the first version of the catalog of heuristics for SoS projects.

Once again, we thank you for your participation in this research project.

Appendix II. Tool code snippet

II.1 JavaScript routine to check model

```
// check model for heuristics and mKAOS semantics
function checkModel() {

    // mensagem de erros a ser gerada
    var msg = '';

    // load nodes properties
    var nodes_txt = document.getElementById("nodes").innerText;
    if (nodes_txt == '') nodes_txt = '[]';

    // parse txt to JSON format
    var arrayNodes = JSON.parse(nodes_txt);

    // counters
    var cntConstituent = 0;
    var cntRefinement = 0;
    var cntMission = 0;

    // nodes loop checking heuristics
    arrayNodes.forEach(function (node) {

        // load properties to test
        var id = node.id;
        var label = node.label;
```

```
var type = node.type;
var interface = node.interface;
var available = node.available;
var checked = node.checked;

// SoS node properties
var SoSType = node.SoSType;
var provider = node.provider;
var builder = node.builder;
var benefits = node.benefits;
var policy = node.policy;

// refinment node
if (type == 'refinement') {
    // just counting yet
    cntRefinement++;
}

// SoS checking
if (type == 'SoS') {

   // not applied to virtual SoS
   if (SoSType !== 'Virtual') {

        // provider defined?
        if (provider == '') {
            msg = msg + "Heuristic IN 1- Identify the
            responsible for providing resources to SoS \n";
        }

        // bulider defined?
        if (builder == '') {
```

```
            msg = msg + Heuristic IN 2- Identify the
            responsible for building and operating
            the SoS \n";
        }

        // who benefits defined?
        if (benefits == '') {
            msg = msg + Heuristic IN 3- Identify who
            benefits from SoS or inform 'nobody' in SoS
            properties \n";
       }

        // feedback policy defined?
        if (policy == '') {
            msg = msg + Heuristic MO 2- Include the
            feedback policy for SoS operation \n";
        }

        // SoS type defined?
        if (SoSType == '') {
            msg = msg + "- Define the SoS type in
            SoS properties \n";
        }
    }
}

// constituent systems checking
if (type == 'constituent') {

    // CS counter
    cntConstituent++;
```

```
            // interface defined?
            if (interface == '') {
                msg = msg + "Heuristic IO 1- No interface
                defined for [" + label + "] \n";
            }
        }

        // missions checking
        if (type == 'mission') {

            // mission counter
            cntMission++;

            // capability not available but checked!!!
            if (available == 'no' && checked == 'yes') {
                msg = msg + "Capability [" + label + "] not
                availabie but checked?\n";
            }

            // capability not available
            if (available == '') {
                msg = msg + "Heuristic SC 1- Define if the
                capability [" + label + "] is availabie \n";
            }

            // capability checked?
            if (checked == '') {
                msg = msg + "Heuristic SC 2- Define if the
                capability [" + label + "] was checked \n";
            }
        }
    }
```

```
    // SoS without minimum elelemts

    if (cntConstituent == 0) {
        msg = msg + "- SoS have no constituent systems \n";
    }
    if (cntMission == 0) {
        msg = msg + "- SoS have no capabilities/missions \n";
    }
    if (cntRefinement == 0) {
        msg = msg + "- SoS have no refinements \n";
    }

    // inform if any issues found

    if (msg == '') {
        swal('Check Model', 'No issues found', 'success');
    } else {
        swal('Check Model', msg, 'warning');
    }

}
```

II.2 mKAOS Studio Lite tutorial

The four pages tutorial included in the developed tool is presented below.

1. Introduction

This tool is intended to assist in the design of systems-of-systems (SoS) projects and was developed within the scope of the Complex Systems Engineering Laboratory (LabESC) at the Federal University of the State of Rio de Janeiro (UNIRIO). This tool uses elements notation from the mKAOS Studio tool, developed using Eclipse environment and built under the Systems Conception Laboratory (ConSiste) at the Federal University of Rio Grande do Norte (UFRN).

SoS are arrangements of systems with operational and managerial independence called constituent systems that unite capabilities to fulfill a new mission that is not the responsibility of any of the constituent systems in isolation. Examples of SoS are smart cities where various public and private systems communicate to promote better urban mobility, emergency care, and health and social welfare.

The SoS mission model diagram, built with this tool, represents the constituent systems involved in SoS design, how these constituent systems participate in SoS and how the capabilities provided are used to generate new capabilities needed to fulfill SoS missions. This diagram can make it easier to communicate project ideas among managers, systems engineers, and developers in the SoS design phase.

1.1 Aobout this tool

This tool is intended to assist in the design of systems-of-systems (SoS) projects and was developed within the scope of the Complex Systems Engineering Laboratory (LabESC) at the Federal University of the State of Rio de Janeiro (UNIRIO). This tool uses elements notation from the mKAOS Studio tool, developed using Eclipse environment and built under the Systems Conception Laboratory (ConSiste) at the Federal University of Rio Grande do Norte (UFRN). This modeling tool was developed in order to assist in the SoS design using rules (heuristics) to be applied during SoS modeling. As the diagram is built, the tool helps both in the composition of the elements, criticizing the types of connections, and infilling of the attributes necessary for improve the chances of success of the project.

It is possible to evaluate possible problems in the model being generated using the [Check Model] button at any time during diagram construction. The messages issued by the tool indicate the possible problem, the element involved, and the respective heuristic involved if any. The tool's own interface also prevents incorrect connections being made between elements, for example, a constituent system A linked to another constituent system B, representing that the system A provides system B, which for SoS does not make sense as constituent systems provide capabilities instead of other systems.

Elements that need additional attributes to be set initially appear with a white background in the diagram. Once this information is entered, the element is filled with the corresponding background color. The figure below shows an example of elements with no attributes indicated (on the left) and with full attributes indicated (on the right). White background notation has been introduced in this tool.

Figure 1. SoS mission model example elements

2. SoS mission model example using mKAOS

Below is an example diagram representing an SoS with two constituent systems providing capability 1 and capability 2. In this model, constituent system 1 (unreliable system) provides capability 1, but it may have temporary failures (clock symbol), and so the redundant system was added to the project to provide a redundant capability to capability 1 in the events of a failure, thus ensuring that the global mission can be fulfilled more reliably. The clock and exclusive gateway notations were introduced in this tool.

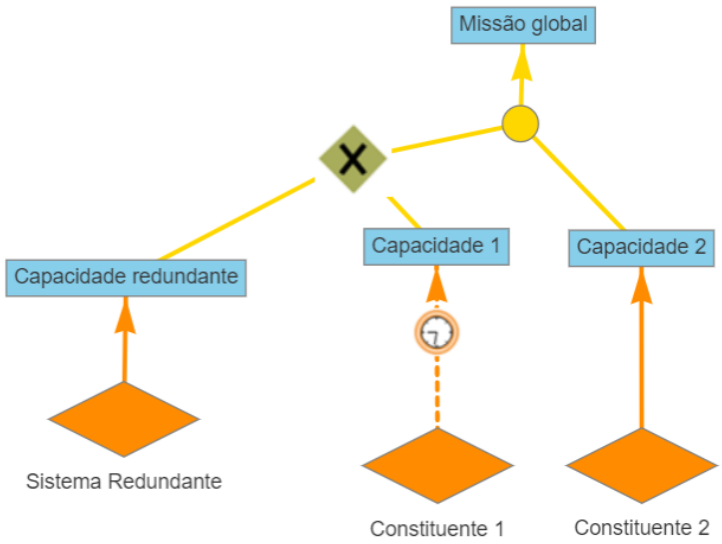


Figure 2. SoS example

3. The elements used in the mission model

The mKAOS mission model diagram uses the following elements:

	Constituent system: Independent system that provides one or more capabilities to the SoS.
	Capability / Mission: Represents a capability provided by a constituent system or a mission accomplished by the SoS.
	Refinement: It is an activity designed to process one or more capabilities provided to deliver new capabilities or perform SoS missions.
	Constituent System Responsible For link: Indicates that a constituent system is responsible for delivering a particular capability.
	Refinement responsible for link: Indicates that a particular refinement is responsible for delivering a new capability or mission.
	Capability provisioning link for refinement: Indicates which capabilities are delivered for refinement. Note that there is no arrow on this edge. as there is no capability or mission delivery for the SoS.

In addition to the above elements, mKAOS Studio Lite adds the following elements:

	Exclusive gateway: used to represent that there is more than one constituent system available to provide redundancy in case of failure to provide some capacity.
	Temporary failure: Happens when a system stops providing a capability but returns to provide it without the need for action, such as when there is a network problem.
	Permanent failure: Occurs when a system stops responding and action is required to that it works again or is replaced, such as when a system update causes an error in a web service..

Below is a schematic of the tool's interface elements:

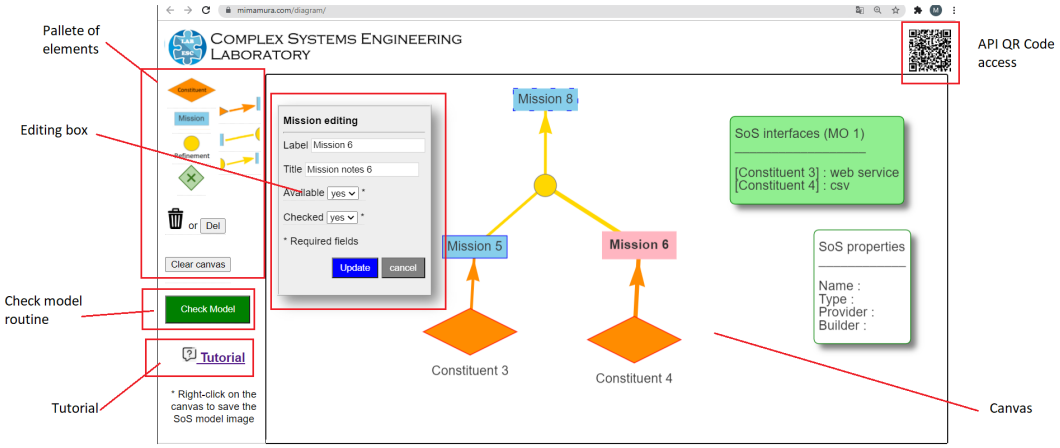


Figure 3. Elements of tool interface

4. Building a mission model in the tool

To build a SoS project using this tool, you can follow these steps:

Step 1. Identify which constituent systems and their capabilities will be used by SoS and create a symbol for each one:

- Click on the  symbol to create each of the constituent systems.
- Click on each of the symbols of the constituent systems and change its name in the "Node" box below the diagram, modifying the generic name given by the tool in the Label field and the hint to be shown when the mouse lands on the figure.
- Identify which capabilities must be combined to generate new capabilities and realize the side missions and the global SoS mission.
- Click on the  symbol to create each of the capabilities. /li>
- Click on each of the capability symbols and change its name in the editing box in the diagram, modifying the generic name given by the tool in the Label field and the hint to be shown when the mouse lands on the figure.
- Link the constituent systems to their respective capabilities by clicking on the symbol  and then in the pair of system and capacity to bind. The tool chooses the correct direction of the arrow.

Step 2. Identify what refinements are needed and what capabilities are involved in each refinement:

- Click on the  symbol to create the representation of the necessary refinements for each of the processes to utilize the capabilities.
- Click on the  symbol and link capabilities to be refined (combined, processed, etc.) by each of the refinements created.
- Click on the  symbol to create the mission to be performed or new capacity being delivered to the SoS.

- For each refinement, click on the  symbol to create the mission to be performed or the new capability to be delivered to the SoS.
- Click on the  symbol and link the refinements to their respective abilities or created missions.

Step 3. If necessary, repeat step 2 to create new refinements to the abilities or quests created by the previous refinements until it was able to represent the global SoS mission.

If you can identify possible flaws in the project already follow step 4.

Step 4. Identify if there are possible failures of the constituent systems to represent them in the model.

- Click on the edge  enter the system that can exhibit failure and your ability to choose the failure type option in the Edge box below the diagram, saving then. An icon representing the type of failure will be added to the link.
- If it is already possible to identify a redundant capacity for this possible failure, create the system and redundant capacity using step 1.
- Click on the  symbol to create the representation alternative consumption in case of failure and link this symbol to the capacities involved.
- - Link  to the respective refinement, denoting that redundant capacity can be consumed in case of main capacity failure.

To delete any element created, just click on it and then click on  icon or [DEL] key.

5 - Heuristics for SoS projects

Heuristics are criteria, key points or "golden rules" that a project needs attend to fulfill a certain proposal. The heuristic can be considered a "mental shortcut" used in human thinking to get to results and questions more complicated quickly and easily, even if they are uncertain or incomplete.

Heuristic evaluation is a method proposed by Jakob Nielsen and Rolf Molich that consists of a inspection to find problems in a user interface. In this tool, we use heuristics to help build SoS projects. They can be checked via the [Check Model] button at the bottom of the toolbar.

Initialization heuristic IN 1 - The project should clearly identify who provides the capabilities and resources necessary for the operation of the SoS.

SoS depends on the synergy between systems that provides the necessary capabilities for its operation, it is necessary to guarantee who supplies them. Example: in an SoS for the prevention of natural disasters, there is a responsible person able to indicate which systems can be integrated to the SoS?

Initialization heuristic IN 2 - The project should clearly identify who is responsible for the construction and operation of the SoS.

When funding is needed for the operation of the SoS, those involved must inform how and by whom this it will be done so that these issues are dealt with at the appropriate time. Example: How much will it cost and who goes to the activities and resources needed to build and maintain the SoS, in addition to those already available by constituent systems?

Initialization heuristic IN 3 - The project should clearly identify who benefits from SoS.

Every SoS is built and operates for a purpose. It is important to identify who are the beneficiaries of the activities or operation of the SoS. Example: who are the possible users of SOS and what is the value of what SoS produces to them?

Constituent systems heuristic CS 1 - Define which capabilities are already available and which they need be implemented in the constituent systems for the construction and operation of the SoS.

The SoS project must anticipate whether the necessary functionalities already exist in the constituent systems or whether it will be necessary to implement new features. Example: it will be necessary to demand that a system of traffic control provides a new capability to produce information to assist on the ambulance path?

Constituent systems heuristic CS 2 - Individual capabilities need to be checked.

Each capability of the constituent systems must be checked to see if it matches what is expected. Example: The designer must verify that the capability of a localization system is available with the frequency and accuracy suitable for the purpose of the SoS.

Interoperability heuristic IO 1 - The interfaces between the constituent systems and the SoS must be defined during the project.

The interfaces between the constituent systems and the SoS are a crucial factor for the operation of the SoS and they are points where the designer can exert influence. Example: The communication required between 2 constituent systems is satisfactorily available for SoS operation or will be required build new channels for connectivity?

Monitoring heuristic MO 1 - The interface patterns that emerged in the evolutionary process must be identified.

The evolutionary process (eg upgrades) may require new communication standards or the updating of projected standards initially. It is necessary to maintain a set of standards used according to the constituent systems and the SoS itself evolve, generating a roadmap for the process. Example: When a new hospital becomes part of the municipal health system, it is necessary to include the communication standards of their systems in the project of the SoS if necessary.

Monitoring heuristic MO 2 - The SoS project should include a feedack policy for the operation of the SoS.

It is necessary to monitor the SoS to detect problems during its operation and define the actions required to deal with them. For example: Is it necessary to monitor the average time of emergency medical care and the respective vacation schedule of professionals to adjust public health activities in a smart city?

6. Questions, suggestions and contact

In case of questions or suggestions, please contact us by email:

marcio.imamura@edu.unirio.br

Thank you for contributing to this study!

Appendix III. Feasibility study

The following research instruments were prepared in portuguese as they were created to be used with groups of researchers in Brazil.

III.1 Itinerary for the feasibility study meetings

Each individual meeting with participants of this study followed the same itinerary:

1st meeting

1. Lecture on SoS

- large number of systems in the modern world
- practically no one else develops systems from scratch
- one of the ways to create new systems is to use ready-made systems
- SoS composed of management and operational independent systems

2. Purpose of the tool and research

- produce a template to communicate SoS design to stakeholders
- evaluate model using heuristics from the literature

3. Simple model explaining the use of the tool

2nd Meeting (1 week later)

1. Ask & answer questions
2. Check if the concepts were absorbed
3. Check if the participant can use the tool
4. Request the completion of the evaluation form if everything is ok

3rd Meeting (if necessary)

- Ask & answer questions about evaluation form if applicable
- Request the completion of the evaluation form again

III.2 Guidelines for tool evaluation

The participants received as the following guidelines prior to attending the first study meeting (in Portuguese):

Orientações para uso e avaliação da ferramenta de modelagem de sistemas-de-sistemas

Esta pesquisa é conduzida por Marcio Imamura (estudante de mestrado do PPGI/UNIRIO) sob orientação do Professor Doutor Rodrigo Santos e colaboração dos mestres Francisco Henrique Ferreira e Juliana Fernandes.

Informações pessoais sobre os participantes não serão divulgadas nos relatórios da pesquisa, que preservará o caráter anônimo e confidencial das respostas. Sua contribuição é extremamente importante para esta pesquisa.

Agradecemos gentilmente sua colaboração!

Marcio Imamura, mestrando (UNIRIO)

Juliana Fernandes (IFPI)

Francisco Henrique Ferreira (UNIRIO)

Rodrigo Santos, professor orientador (UNIRIO)

Em caso de dúvidas ou pedido de informações extras, por favor, entre em contato por email: marcio.imamura@edu.unirio.br

Seção 1. Termo de consentimento livre esclarecido (TCLE)

Ao responder a este questionário, você permite que os pesquisadores obtenham, usem e divulguem as informações geradas a partir dos dados agrupados conforme descrito abaixo. CONDIÇÕES

1. Eu entendo que todas as informações são confidenciais. Eu não serei pessoalmente identificado e concordo em concluir o questionário para fins de pesquisa. As informações derivadas dessa pesquisa anônima podem ser publicados em periódicos, conferências e publicações em blogs.

2. Entendo que minha participação nesta pesquisa é totalmente voluntária e que recusar participar não envolverá penalidade ou perda de benefícios. Se eu escolher, posso retirar minha participação a qualquer momento. Eu também entendo que, se eu optar por participar, posso me recusar a responder questões abertas as quais eu não me sinta confortável.

3. Entendo que posso entrar em contato com o pesquisador se tiver alguma dúvida sobre a pesquisa. Estou ciente de que meu consentimento não me beneficiará diretamente. Também estou ciente de que o autor manterá os dados de maneira agrupada, coletados em perpetuidade e poderá utilizá-los para trabalhos acadêmicos futuros.

4. Ao prosseguir para a próxima seção, eu livremente reconheço meus direitos como participante voluntário(a) da pesquisa, conforme descrito acima, e forneço consentimento ao pesquisador para usar meus dados na condução de pesquisas sobre a área mencionada acima.

Seção 2. Sobre sistemas-de-sistemas

Sistemas-de-sistemas (SoS) podem ser compreendidos como um arranjo de sistemas independentes (gerencial e operacionalmente), interagindo para cumprir uma nova missão, que não é cumprida por nenhum dos sistemas constituintes isoladamente.

Um exemplo de SoS são as cidades inteligentes, onde vários sistemas públicos e privados colaboram para melhorar a qualidade de vida dos cidadãos e otimizar o uso de recursos públicos. Neste exemplo, sistemas independentes como os sistemas de trânsito podem colaborar com os sistemas de hospitais e da defesa civil para coordenar atendimento de emergência em caso de desastres naturais e outras tragédias de grande porte.

De maneira geral, existem dois tipos de missões em SoS: globais, sendo as missões que devem ser realizadas pelo SoS como um todo e individuais (ou capacidades) de cada sistema constituinte que são refinadas (combinadas, transformadas etc) para que o SoS possa cumprir sua(s) missão(ões) globais.

O software a ser avaliado neste trabalho utiliza a notação do modelo de missões mKAOS para retratar o relacionamento entre os sistemas constituintes para entregar a missão do SoS, deixando de lado qualquer informação relacionada a implementação específica dos sistemas constituintes ou como a comunicação entre eles deverá ser implementada.

A ferramenta mKAOS foi produzida no Laboratório de Concepção de Sistemas (ConSiste) do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte. Os elementos gráficos da notação são descritos

abaixo:

Sistemas constituintes: Sistemas que fornecem capacidades ao SoS

Missão ou capacidade: No caso dos constituintes, é a capacidade fornecida ao SoS ou missão do SoS quando resultado de refinamento.

Refinamento: Processo que compila uma ou mais capacidades fornecidas em uma missão do SoS.

Link "responsável por": Indica qual sistema constituintes ou refinamento é responsável pelo fornecimento da capacidade ou missão.

Link sem setas: indica entrega de capacidades para consumo do refinamento.

Seção 3. Descrição do problema

Para este exercício de avaliação do software em questão, considere a seguinte situação que ocorre no Instituto Brasileiro de Geografia e Estatística:

Nesse instituto, existem vários sistemas técnicos e administrativos que em certos momentos acabam fornecendo capacidades para cumprir o objetivo de planejar novas contratações de profissionais bem como subsidiar pedidos de recursos para realização das atividades de coleta de informações.

Os Sistemas Constituintes

Os sistemas técnicos cuidam das pesquisas e fornecem a informação para processamento estatístico e controlam a quantidade de questionários coletados pelas agências do IBGE em todo Brasil. Os 4 sistemas técnicos envolvidos são: Sistema de pesquisas agropecuárias. Coletam informações sobre produção dos estabelecimentos agropecuários como, por exemplo, a produção de grãos, leite, ovos, tomates, laranja etc.

Sistema de pesquisas econômicas. Estes sistemas controlam a coleta de informações sobre atividades econômicas, como comércio e serviços. Sistema do registro civil. São realizadas junto aos cartórios, coletando informações sobre nascimentos e mortes.

Sistema da pesquisa nacional por amostra de domicílios. Realiza coleta de informações sobre a população como trabalho e rendimento.

Além de controlarem a distribuição dos questionários e do andamento da coleta de informações, juntos estes sistemas fornecem o total de questionários coletados.

Os sistemas administrativos cuidam de atividades necessárias para apoiar as

atividades técnicas possibilitando alocação adequada de recursos humanos e materiais envolvidos na coleta, observando o cumprimento das normas legais e boas práticas do governo federal. Os 3 sistemas administrativos envolvidos são:

Sistema para trabalhadores permanentes. Controla pessoal efetivo alocado na sede da instituição e em coordenação de trabalhadores temporários contratados nas agências do IBGE para coletar questionários.

Sistema para trabalhadores temporários. Contratados para fazer a coleta nas agências do IBGE, realizando a tarefa de ir a campo para coletar informações Sistema para controle de viagens. Controla diárias, passagens e utilização de veículos quando são necessários estes tipos de recursos para realização das pesquisas.

Além de observarem todo o aspecto legal e do processo de utilização dos recursos humanos e materiais, estes sistemas fornecem as informações sobre pessoal disponível e viagens a serviço. Objetivo

Dada a situação descrita, modele o SoS utilizando a ferramenta para representar como os sistemas constituintes interagem para entregar a missão global do SoS de fornecer informações para o planejamento e contratação de recursos humanos e materiais com o objetivo de viabilizar a coleta de questionários de forma econômica e racional.

A ferramenta pode ser acessada no link:

http://mimamura.com/diagram

Pedimos que a ferramenta seja utilizada por cerca de 1 semana ou o tempo necessário para conhecê-la, utilizá-la e tirar suas dúvidas. Após sua experiência, por favor utilize o seguinte link para fazer a avaliação.

https://forms.gle/kJKZxCzKVrBh6iVe8

Qualquer dúvida entre em contato pelo e-mail marcio.imamura@edu.unirio.br

Obrigado!

III.3 Data collection form

The 7 pages of the data collection form for the feasibility study survey are presented in the next pages with dummy marks on the questions (in Portuguese).

Avaliação da ferramenta de modelagem para projetos de sistemas-de-sistemas

Esta pesquisa de opinião é conduzida por Marcio Imamura (estudante de mestrado do PPGI/UNIRIO) sob orientação do Professor Doutor Rodrigo Santos e colaboração dos mestres Francisco Henrique Ferreira e Juliana Fernandes.

O preenchimento levará aproximadamente 10 minutos.

Agradecemos gentilmente sua colaboração!

Marcio Imamura, mestrando (UNIRIO)
Juliana Fernandes (IFPI)
Francisco Henrique Ferreira (UNIRIO)
Rodrigo Santos, professor orientador (UNIRIO)

Em caso de dúvidas ou pedido de informações extras, por favor, entre em contato por email:
marcio.imamura@edu.unirio.br

TERMO DE CONSENTIMENTO LIVRE ESCLARECIDO (TCLE)

Ao responder a este questionário, você permite que os pesquisadores obtenham, usem e divulguem as informações geradas a partir dos dados agrupados conforme descrito abaixo.

CONDIÇÕES

1. Eu entendo que todas as informações são confidenciais. Eu não serei pessoalmente identificado e concordo em concluir o questionário para fins de pesquisa. As informações derivadas dessa pesquisa anônima podem ser publicados em periódicos, conferências e publicações em blogs.

2. Entendo que minha participação nesta pesquisa é totalmente voluntária e que recusar participar não envolverá penalidade ou perda de benefícios. Se eu escolher, posso retirar minha participação a qualquer momento. Eu também entendo que, se eu optar por participar, posso me recusar a responder questões abertas as quais eu não me sinta confortável.

3. Entendo que posso entrar em contato com o pesquisador se tiver alguma dúvida sobre a pesquisa. Estou ciente de que meu consentimento não me beneficiará diretamente. Também estou ciente de que o autor manterá os dados de maneira agrupada, coletados em perpetuidade e poderá utilizá-los para trabalhos acadêmicos futuros.

4. Ao seguir para a próxima seção, eu livremente, reconheço meus direitos como participante voluntário(a) da pesquisa, conforme descrito acima, e forneço consentimento ao pesquisador para usar meus dados na condução de pesquisas sobre a área mencionada acima.

O questionário será apresentado após o aceite deste TCLE. *

- [x] Aceito as condições descritas no TCLE.

Perfil do participante

Email (opcional) *

abc@def.ghi

Há quanto tempo você trabalha com sistemas de informação? *

- ○ 0 a 5 anos
- ○ entre 5 e 10 anos
- ◉ mais de 10 anos

Qual sua ocupação atual? *

- ○ Gerente
- ○ Desenvolvedor
- ○ Suporte
- ◉ Outra

Qual seu grau de formação *

- ○ Ensino médio
- ○ Graduação
- ○ Especialização/MBA
- ○ Mestrado
- ◉ Doutorado

Qual nível de conhecimento em modelagem de sistemas? *

- ○ Nenhum
- ○ Pouco
- ○ Médio
- ○ Bom
- ◉ Muito bom

Qual nível de conhecimento em SoS? *

- ○ Nenhum
- ○ Pouco
- ○ Médio
- ○ Bom
- ◉ Muito bom

Avaliação da ferramenta

Avaliação da facilidade de uso da ferramenta

Com base na utilização da ferramenta responda às questões abaixo

1) Foi fácil aprender a utilizar a ferramenta. *

- ○ Discordo totalmente
- ○ Discordo
- ○ Indiferente (ou neutro)
- ○ Concordo
- ◉ Concordo totalmente

2) Consegui utilizar a ferramenta da forma como eu gostaria. *

- ○ Discordo totalmente
- ○ Discordo
- ○ Indiferente (ou neutro)
- ○ Concordo
- ◉ Concordo totalmente

3) Entendi o que estava acontecendo durante a interação com a ferramenta. *

- ○ Discordo totalmente
- ○ Discordo
- ○ Indiferente (ou neutro)
- ○ Concordo
- ◉ Concordo totalmente

4) Consegui executar as tarefas facilmente com o uso da ferramenta, *

- ○ Discordo totalmente
- ○ Discordo
- ○ Indiferente (ou neutro)
- ○ Concordo
- ◉ Concordo totalmente

Avaliação da utilidade da ferramenta

Com base na utilização da ferramenta responda às questões abaixo:

5) Acredito que a utilização da ferramenta com as heurísticas foi útil para representar o SoS na situação proposta. *

- ○ Discordo totalmente
- ○ Discordo
- ○ Indiferente (ou neutro)
- ○ Concordo
- ◉ Concordo totalmente

6) A utilização da ferramenta permitiu compreender como os sistemas constituintes se relacionam e em que pontos podem haver problemas para alcançar a missão global do SoS. *

- ○ Discordo totalmente
- ○ Discordo
- ○ Indiferente (ou neutro)
- ○ Concordo
- ◉ Concordo totalmente

7) A utilização das heurísticas na ferramenta melhorou meu desempenho na execução das tarefas propostas. *

- ○ Discordo totalmente
- ○ Discordo
- ○ Indiferente (ou neutro)
- ○ Concordo
- ◉ Concordo totalmente

8) A utilização da ferramenta dá suporte às atividades de gerenciamento de TI *

- ○ Discordo totalmente
- ○ Discordo
- ○ Indiferente (ou neutro)
- ○ Concordo
- ◉ Concordo totalmente

Comentários e sugestões

De acordo com sua opinião, foram identificados aspectos positivos ou negativos da utilização da ferramenta? Se sim, qual(ais)?

Você possui alguma sugestão de melhoria para a ferramenta ou da aplicação das heurísticas? Em caso positivo, por favor especifique-a.

Este espaço é reservado para quaisquer comentários adicionais (dificuldades, críticas e/ou sugestões) a respeito do estudo executado. Contamos com sua contribuição para que o trabalho seja aprimorado.

Obrigado pela participação neste estudo!

III.4 Survey responses to Q9, Q10 and Q11

All participants responses to Q9, Q10 and Q11 questions are presented in the next pages (in Portuguese).

Q9. De acordo com sua opinião, foram identificados aspectos positivos ou negativos da utilização da ferramenta? Se sim, qual(ais)?

Positivo : bem fácil e intuitiva.

Negativo : a opção de "apagar" um elemento.

Posivito: Permitiu observar a fragilidade do sistema em seu estado atual. Percebendo que falhas nos níveis mais baixos da cadeia de sistemas, impede o funcionamento por completo do sistema.

Negativo: 1) A dimensão de altura disponível para visualização é pequena e poderia ser aumentada.

2) Os novos nós são criados numa posição fixa, que, dependendo do tamanho do diagrama e do local sendo visualizado, fica escondida.

Positivo: amigável, interface simples e enxuta, bom tempo de resposta, e independente de plataforma (basta ter um navegador). Negativo: documentação pode ser melhorada, principalmente no aspecto conceitual. Também senti falta de um exemplo prático.

Considerei a ferramenta de fácil utilização, com interface simples e intuitiva.

Também considerei a indicação gráfica de inconsistêncis nos atributos (quando a figura fica cheia ou vazia) muito importante para rápida visualização do estado do SoS. A ferramenta de verificação de consistências no modelo é de grande ajuda para indicar exatamente o que deve ser corrigido.

Pontos positivos: facilidade de uso, de edição de propriedades, interface limpa

Aspectos positivos:

1) utilização de cores e preenchimento com significado; menu lateral com os símbolos ajudou bastante; aplicação e criação de nós na modelagem foi bastante intuitivo;

2) pareceu ser uma ferramenta simples e eficaz para criar um modelo conceitual, as explicações no tutorial esclareceu o seu uso e o exemplo prático facilitou bastante nisto;

3) os campos obrigatórios evitaram o excesso de mensagens heurísticas e direciona

o usuário para o que deve ser realizado;

4) de maneira geral, gostei de modelar com o software e usaria-o em projetos, como um módulo ou plugin de modelagem;

Aspectos negativos:

1) as propriedades do sistema poderiam ser citadas no tutorial (Provider / Builder);

2) no tutorial poderia aparecer o uso do available/check;

3) talvez seja o caso das propriedades do SoS pertencerem ao canvas, não a um objeto desenhado, a não ser que as missões gerais estejam ligadas a ele;

Achei a ferramenta bem intuitiva e de fácil utilização, com destaque para as trocas de cores em relações as ações que são executadas, como por exemplo um elemento que precisa preencher um determinado atributo fica com o fundo claro. Outro ponto positivo da ferramenta são as próprias validações feitas pela mesma, antes mesmo de validar o modelo, a ferramenta não permite que conexões que não fazem sentido para um SoS sejam modeladas. Existem ainda outros pontos positivos como o tutorial que no entanto não foi nem tanto necessário devido a facilidade de utilização da ferramenta. Por fim achei interessante a utilização do QR para geração da modelagem em formato Json permitindo que outros usuários remotos possam editar em conjunto.

Q10. Você possui alguma sugestão de melhoria para a ferramenta ou da aplicação das heurísticas? Em caso positivo, por favor especifique-a.

Poderíamos avaliar a experiência do usuário com relação a usabilidade e utilidade da ferramenta.

1) Aumentar a altura da caixa em que é gerado o diagrama;

2) Criar os nós sempre num local visível da tela, como no canto superior esquerdo;

3) O ícone do Exclusive gateway lembra muito botões de Cancelamento ou Exclusão, eu trocaria para um random icon.

Sugiro aprimorar a documentação, principalmente no aspecto conceitual. O tutorial apresentado apresenta exemplo e explicações muito abstratas, distantes de uma aplicação prática.

Para melhorar a usabilidade da ferramenta, sugiro que, ao adicionar um item, este seja adicionado na visualização atual (em um grande diagrama, o item se perde)

Para faciliar a leitura do estado atual do SoS, sugiro adicionar indicadores gráficos para os atributos da Missão (Avaliable e Checked).

Também para facilitar a leitura da configuração atual do SoS, sugiro criar um descritivo dos itens onde conste a interface dos sistemas constituintes.

Sugestão: um botão de desfazer. Às vezes eu me arrependia de uma ação ou apagava acidentalmente um componente.

Melhorias:

1) possibilidade de salvar o modelo para poder dar continuidade na modelagem;

2) o uso de heurísticas foi bastante válido, talvez seja possível otimizar suas sugestões por algum linguagem logica (por exemplo, Prolog muito usado no campo de IA, fazendo uso de logica de programação com regras e predicados);

3) trechos do tutorial poderia aparecer opcionalmente em cada mensagem heurísitca, afim de evitar que o usuário abra uma página à parte para ler o detalhe de cada heurística;

4) a tela para modelagem está em inglês, mas o tutorial somente em português, poderia haver uma caixa de seleção para linguagem desejada ser a mesma em ambas;

5) poderia ser útil adicionar uma tela para cadastro das interfaces (com campo de descrição opcional), para que o usuário não tenha que repetir a digitação manual das interfaces toda vez que quiser modelar um novo SoS;

6) a interface do programa pode ser mais bonita (de maneira geral) em versões mais avançadas;

1 - Ao utilizar a ferramenta um popup do JavaScript foi acionado, seguinte mensagem de erro: "TypeError: Cannot read property '1' of undefined".

2 - Ao clicar duas vezes em um objeto em tela é disparado uma mensagem "a".

3 - O gateway ainda está com um fundo branco.

4 - Acho que expandir a ideia do Qr para uma edição em conjunto seria muito interessante, não me recordo de muitas ferramentas próprias para modelagem com essa funcionalidade.

5 - Ao clicar no check model várias heurísticas são validadas, acredito que fosse interessante dividir essas heurísticas, como se fossem warnings e erros. Algumas heurísticas o projeto não builda com outras o usuário é apenas informado sobre o problema.

6 - Ao pensar em herísticas me veem a mente a ideia de métricas, daria ainda para elaborar como trabalho futuro algo que desse um grau de confiança para o modelo.

7 - Na área de modelagem as pessoas estão muito preocupadas em relação como o modelo vai ficar, no entanto, já existem algumas ferramentas que fazem como se fosse uma execução do modelo. Acho que seria interessante poder modelar o SoS e ver algumas interações entre os modelos. O próprio camunda (ferramenta para modelagem) possui um plugin para execução de processos para BPMN,

Q11. Este espaço é reservado para quaisquer comentários adicionais (dificuldades, críticas e/ou sugestões) a respeito do estudo executado. Contamos com sua contribuição para que o trabalho seja aprimorado.

1) Ao deixar uma capacidade sem sistema responsável, o Check não identificou isso como problema. Havia entendido pela IN 1 que isso seria verificado;

2) Poderia ter uma heurística para verificas a existencia de Capacidades com mais de 1 Constituinte;

3) Entendo que arestas com possibilidade de falhas permanentes e temporárias poderiam aparecer no Check Model;

Senti falta de alguma explicação sobre os conceitos e fundamentos de SoS na ajuda da ferramenta. A explicação disponível em http://mimamura.com/diagram/tutorial/ até apresenta um exemplo no item 2, mas não a explicação lá presente abordava de forma muito vaga essa questão conceitual. Diferente de uma ferramenta de modelagem de banco de dados, em que sei por exemplo o conceito de entidade, relacionamento, chave primária, chave estrangeira, e relacionamentos. Mas aqui, no caso, não domino os conceitos que envolvem sistemas constituintes, capacidades e refinamentos, e de que forma eu posso usar os mesmos e compreender melhor o que eu estou modelando com a ferramenta. Também a explicação sobre a heurística no item 5 do tutorial não ajudou muito em compreender de que forma poderia ser usada a ferramenta em uma aplicação prática.

A fim de buscar mais sobre estes conceitos, fiz uma busca pelo assunto na internet, e usei os seguintes documentos como referência, que me ajudaram a entender um pouco melhor o que poderia ser feito com essa ferramenta.

Deixo aqui o link para estes documentos:

https://edisciplinas.usp.br/pluginfile.php/4475821/mod$_r esource/content/1/Aula9_S istemaDeSi$
Resumida.pdf

http://www.dimap.ufrn.br/ everton/publications/2015-SESoS-mKAOS.pdf

http://www.dimap.ufrn.br/ everton/publications/2017-SESoS-M2Arch.pdf

Nos links acima há exemplos práticos que poderiam ser usados como referência para construção do tutorial de uso (mostrando passo a passo quais constituintes, capacidades e refinamentos estão sendo inseridos na modelagem, explicando qual a finalidade de cada um dentro do SoS, de forma a tornar a explicação não tão abstrata e mais próxima de uma aplicação da vida real).

Quanto à notação, na minha percepção o símbolo de sistema constituinte remete ao símbolo de decisão do BPMN, causando certa confusão em um primeiro momento. Talvez a escolha de outro elemento gráfico - se possível - pudesse ajudar na leitura do gráfico.

Comentário: label e title me confundiam, porque parecem descrever a mesma coisa. Title poderia ser "hint" ou "notes".

Modelagem simples de ser feita, objetivando os pontos prioritários de um SoS, o uso de heurísticas foi bem útil para a correta modelagem, pode-se pensar em aprimorar o software e integra-lo com outras ferramentas de modelagem, tal como é feito com softwares que trabalham com RUP e UML, onde os modelos conversam com outras ferramentas de projeto; Num momento oportuno, seria interessante pensar em portabilidade do modelo;

Achei bastante interessante o trabalho, interface muito intuitiva e fácil aprendizado. Não sou um pleno conhecedor sobre SoS, no entanto, a partir da ferramenta consegui modelar o exemplo de SoS sem grandes dificuldade.