A Framework for the Design of Fault-Tolerant Systems-of-Systems

Francisco Henrique Cerdeira Ferreira[a], Elisa Yumi Nakagawa[b], Antonia Bertolino[c], Francesca Lonetti[c], Vânia de Oliveira Neves[d], Rodrigo Pereira dos Santos[a]

[a] *Federal University of the State of Rio de Janeiro (UNIRIO), Rio de Janeiro, Brazil*
[b] *University of São Paulo (USP), São Carlos, Brazil*
[c] *Institute of Information Science and Technologies (ISTI-CNR), Pisa, Italy*
[d] *Fluminense Federal University (UFF), Niterói, Brazil*

Abstract

Context: Systems-of-Systems (SoS) increasingly permeate everyday life in various critical domains. Due to their dynamic nature, guaranteeing their fault tolerance is challenging. Fault-tolerant SoS must deal with behavioral changes in constituent systems, whether accidental or deliberate.

Goal: This work proposes ReViTA, a framework to assist professionals in designing fault-tolerant SoS that can continue to provide their function even in the presence of disturbances, i.e., events that affect the ability of an SoS to fulfill its mission.

Method: By adopting ReViTA, fault tolerance can be achieved by reconfiguring an SoS architecture to meet the critical mission requirements.

Results: We performed two studies to evaluate the ReViTA acceptance by professionals. In the former, we gathered perceptions and suggestions from 14 professionals through individual interviews. In the latter, we involved a group of four professionals who applied ReViTA to a real-world scenario.

Conclusions: The results demonstrate that ReViTA can effectively support professionals in designing fault-tolerant SoS. Employing ReViTA also brings insights into costs and planning that are crucial for implementing fault-tolerance strategies. Using ReViTA facilitates a comprehensive understanding of conflicts and weaknesses in constituent systems and fosters collaboration between domain experts and decision-makers. Employing ReViTA also improves stakeholder communication and enhances resource utilization.

Email addresses: francisco.ferreira@uniriotec.br (Francisco Henrique Cerdeira Ferreira), elisa@icmc.usp.br (Elisa Yumi Nakagawa), antonia.bertolino@isti.cnr.it (Antonia Bertolino), francesca.lonetti@isti.cnr.it (Francesca Lonetti), vania@ic.uff.br (Vânia de Oliveira Neves), rps@uniriotec.br (Rodrigo Pereira dos Santos)

1. Introduction

In recent years, we assist to the emergence of large-scale, complex, software-intensive systems in various domains, including aerospace, defense, and healthcare, also facilitated by the proliferation of disruptive technologies such as Cloud Computing, Artificial Intelligence, and the Internet of Things (IoT). These large-scale complex systems are often composed of independent systems mutually interacting to achieve broader goals, which are commonly referred to as systems-of-systems (SoS) [1].

SoS are complex networks of constituent systems, each capable of operating independently, that contribute towards achieving, when interconnected, some common goals – or missions – surpassing the capabilities of each single constituent system operating in isolation [1]. The complexity of SoS arises from the diversity of constituent systems, their capacity to evolve independently, and their interactions. Such interactions can lead to emergent behaviors at the SoS level, a phenomenon that cannot be predicted from the individual behaviors of the constituent systems [2].

While SoS hold significant potential in addressing the complex challenges of the modern world, they come with inherent risks. Given their widespread adoption and the critical importance of the domains in which they operate, it is crucial to ensure an adequate level of reliability [3]. The lack of reliability could lead to severe consequences, such as environmental damage, economic losses, injuries, and even loss of human lives [4]. In healthcare, for instance, the lack of reliability of an SoS managing patient data can result in inaccuracies or loss of critical patient information, thereby threatening patient safety and treatment effectiveness [5]. As another example, a problem in a traffic management SoS could lead to traffic congestion or accidents [6].

In traditional systems, reliability is commonly associated with component failures [7]. In SoS, reliability is also affected by any event, whether deliberate or unintentional, that affect the ability of an SoS to fulfill its mission [3]. These events are referred to as "**disturbances**" [8]. These include failures and undesirable behaviors of constituent systems. A significant challenge arises when these undesirable behaviors, while acceptable in the standalone context of an individual constituent system, become harmful within the SoS context. This occurs due to the independence of the constituent systems, which are often managed by different organizations and operate in diverse environments [9], which may lead to conflicts SoS goals [3, 10].

As it may be challenging to control the disturbances in SoS due to the independence of the constituent systems, our focus is on fault tolerance, i.e., strategies that ensure SoS continue to provide their function even in the presence of disturbances. By focusing on fault tolerance, we aim to mitigate the impact of disturbances and enhance the overall SoS reliability, ensuring the ability of SoS to fulfill their missions and produce desired effects.

While the existing literature on SoS fault tolerance has advanced in the last years [3], critical aspects still require investigation. In particular, a significant gap exists regarding the SoS dynamic nature. Constituent systems can produce

disturbances at the SoS level that might result from failures, implementation changes, or, under certain circumstances, deviations from their roles within the SoS due to competing objectives. In addition, **constituent systems can produce temporary disturbances as they undergo behavioral changes in response to specific environmental stimuli and return to their normal behavior after a while** [11]. Existing literature has not adequately addressed this additional layer of dynamism to the best of our knowledge. Hence, the real-world problem is that such dynamism and unpredictability can affect the overall SoS reliability, making it challenging to maintain their ability to accomplish missions successfully.

In response to this challenge, we introduce ReViTA (**Re**configurations **Vi**a **T**ransient **A**rchitectural Configurations). ReViTA is a prescritive framework designed to enhance fault tolerance in SoS. This framework provides a structured approach to assist professionals in designing fault-tolerant SoS through architectural reconfigurations, which consists of changing an SoS architecture to adapt to new conditions. These reconfigurations involve modifying SoS composition and relationships. Unlike studies employing SoS reconfigurations to mitigate the impact of disturbances, ReViTA harnesses the opportunistic nature of SoS design to leverage fault tolerance. This is vital because it considers the inherent flexibility of SoS design, offering a more reasonable approach to handling disturbances.

We performed two studies to evaluate the feasibility of ReViTA, involving semi-structured interviews with professionals with different backgrounds. In the first study, we presented the framework to 14 professionals to gather their perceptions and suggestions on the framework through individual interviews. The second study involved a group of four professionals who applied ReViTA using a real-world scenario, specifically an SoS designed to respond to power outages at a large Brazilian public university with distributed institutes and campuses. This last study focused on the professionals' experiences using ReViTA and the potential impact of the framework on their work. The results indicate the acceptance of ReViTA by professionals. The evaluations further unveil the ReViTA's potential in facilitating stakeholder communication and optimizing resource utilization. Furthermore, our research highlights the needs for domain experts and decision-makers to engage in discussion concerning SoS fault tolerance. Their involvement deepens the understanding regarding potential weaknesses and conflicts of fault tolerance strategies. They also contribute with essential insights related to costs, resources utilization, and strategic planning. Our findings reveal that these aspects are vital for successfully enable fault tolerance in SoS.

Summarizing, the main contributions of this study are:

- The introduction of ReVITA, a prescriptive framework designed to enhance fault tolerance in SoS;
- The introduction of the concept of Transient Architectural Configurations (TAC) to mitigate the impact of disturbances in SoS;

- A first evaluation of the ReVITA feasibility by semi-structured interviews with 14 professionals to collect their perceptions of the framework from an intuitive perspective;
- A second evaluation of the ReVITA feasibility with a group of 4 professionals who applied ReViTA in a large Brazilian real-world scenario and provided their opinion from a practical perspective; and
- Finally, a discussion of the main findings and implications of ReVITA usage by researchers and practitioners.

The remainder of this article is organized as follows: Section 2 presents the background whereas Section 3 shows related work;Section 4 details our research method; Section 5 introduces ReViTA; Section 6 presents the results of the evaluation conducted through two distinct studies; Section 7 presents a discussion of the results, implications for practitioners and researchers, and lessons learned; Section 8 discusses the threats and limitations of this work; and Section 9 concludes this article.

2. Background

This section lays the foundation for understanding the key concepts relevant to our study: SoS, fault tolerance, systems reconfiguration, and self-adaptation.

2.1. System-of-Systems

An SoS is a network of independent systems, known as the constituent systems, that are connected to achieve a common goal. The constituent systems are managed by different organizations, using different technologies, and with different operational objectives. The synergistic interaction among the constituent systems is essential to the overall functioning of the SoS [1].

SoS are becoming increasingly common in our daily life. For instance, in healthcare, an SoS integrates a range of constituent systems, including electronic health records systems, medical imaging systems, and patient monitoring systems to improve the efficiency and quality of patient care [5]. In the domain of urban mobility, SoS encompasses urban traffic management, control of intelligent traffic lights, and integrated public transportation systems to optimize traffic flow and enhance urban mobility [6].

According to Mark Maier [1], the main characteristics of SoS are: (i) **Operational Independence**: It denotes that constituent systems can function and achieve their objectives independently of the SoS as a whole; (ii) **Managerial Independence**: It indicates that each constituent is managed individually, rather than being centrally controlled; (iii) **Evolutionary Development**: This implies that an SoS can evolve over time, responding to changes in its environment, constituent systems, or objectives; (iv) **Distribution**: It refers to the physical decoupling of constituents within the SoS, necessitating a communication channel for information exchange among these them; and (v) **Emergent**

Behavior: It suggests that the behavior of SoS as a whole emerges from the synergistic interaction among its constituent systems, leading to outcomes that could not be predicted based solely on the properties of the constituent systems.

Moreover, SoS can be categorized according to the levels of authority over the constituent systems, as in the following [12, 13]: (i) **directed**: SoS are centrally managed by an authority responsible for driving operations; (ii) **acknowledged**: SoS have recognized objectives, a controller, and resources at the SoS level, but the controller has no complete authority; (iii) in **collaborative** SoS, there is no central control and the constituent systems work together to fulfill agreed purposes; and (iv) in **virtual** SoS, there is no managerial authority and no commonly established goals.

SoS exhibits an inherent dynamism, which is a natural consequence of the independence of constituent systems. This means that the constituent systems can change at runtime [14], affecting the SoS overall behavior. Such dynamism implies that a range of potential behaviors, both desired and undesired, can emerge from the interactions among constituent systems. This characteristic, in particular, raises considerable concerns about SoS reliability. The changes in constituent systems and their interconnections, driven by both environmental adaptations and evolutionary development, underscore effective **SoS reconfigurations** [15]. It is essential to determine the current and intended future state of SoS. This involves changes in an SoS architecture by modifying its composition of and interconnections among constituent systems [16].

This architectural adaptability, however, is intricately tied to the SoS type, and it can affect the degree of cooperation among the constituent systems. This can influence reconfigurations, such as the manner and timing of the reconfiguration operations. For instance, one approach to reconfiguration involves temporarily placing constituent systems into a "passive" state, i.e., they cease providing services [16]. This approach may be viable for directed SoS, in which a central authority strongly influences the constituent systems. However, in collaborative SoS, this may not be feasible since there is no authority, and the functioning of these types of SoS is primarily based on agreements.

2.2. Fault Tolerance

Fault tolerance is an important characteristic of reliable systems [17]. It entails a system's ability to maintain and deliver a desired level of functionality even in the presence of faults. Ensuring fault tolerance is essential for the uninterrupted operation of systems. Typically, it is achieved through a combination of **error detection** and subsequent **system recovery** [18, 19]. While error detection is designed to generate an error signal or message within the system, system recovery aims to change a system to a state free from errors and faults, thereby allowing a system to be reactivated [18].

Employing redundancy is one of the means to achieve fault tolerance as it ensures that if a component fails, an appropriate backup is in place to take over its role, enabling the system to maintain its operation [18, 20]. The most common form of redundancy involves replacing a failed component with an identical one. On the other hand, when redundancy is applied at a functional

level — replacing a failed component with a different one that performs a similar function — it is called **heterogeneous redundancy** [21]. For instance, if a system that uses push notifications fails, the system could switch to sending email notifications or Short Messaging Service as an alternative. Though email and Short Messaging Service are different communication channels than push notifications, they still fulfill the notification function.

2.3. Systems Reconfiguration

Reconfiguration is a broad term that can apply to various types of systems, including computer systems, network systems, and manufacturing systems, to mention a few. In general, reconfiguration refers to the process of changing the setup or composition of a system [22, 23]. This can involve changing hardware, software, or other system components to improve performance, add new functionality, or adapt to changing conditions.

2.4. Self-Adaptation

Self-adaptation refers to the ability of a system to dynamically adjust its behavior without external intervention. This characteristic is crucial for dealing with dynamic and complex environments, in which operational conditions can change rapidly. Self-adaptive systems continuously monitor their performance and the surrounding context, using gathered information to optimize their operation [24]. This approach is particularly valuable in SoS, since it is essential to address uncertainties and variations in operational conditions.

To enable fault tolerance, SoS must possess self-adaptive capabilities. Addressing the challenges associated with self-adaptation involves the development of effective methods for modeling, analyzing, and designing mechanisms that ensure the quality attributes of SoS [25]. An influential approach frequently utilized in self-adaptive systems is the MAPE-K model [26], comprising five essential components: Monitor, Analyze, Plan, Execute, and Knowledge. The MAPE-K relies on feedback loops to continuously monitor the system's behavior and its environment, analyze collected data, formulate appropriate actions for performance maintenance or enhancement, and subsequently execute these plans.

3. Related Work

Three primary topics are relevant to our research: fault tolerance, SoS reconfigurations and self-adaptation. In this section, we discuss the related work to these subjects.

Regarding SoS fault tolerance, few studies address the topic. Among them, we highlight the work of Andrews *et al.* [27], which presented a disciplined approach for modeling fault-tolerant SoS using SysML. The approach is based on a separation of normal and erroneous behavior of SoS. It supports reasoning about SoS faults and errors, error propagation, and fault and error handling in

the SoS architecture. This work was later extended to enable the translation of SysML models into a formal notation [28], allowing for more rigorous modeling and verification of the SoS architecture concerning fault tolerance.

Andrews *et al.* [29] also introduced the Fault Modeling Architectural Framework (FMAF), a structured method for capturing requirements for fault-tolerant SoS. FMAF supports the development of fault-tolerant architectures and provides a traceable mapping of fault-tolerace requirements into SoS architectural designs. Ingram *et al.* [30] presented an example of the application of FMAF in a traffic management SoS and discussed potential extensions to the framework.

Some studies employed alternative constituent systems as heterogeneous redundancies to compensate for failures or the low performance of primary constituent systems. Uday and Marais [14] introduced the "stand-in redundancy" concept, offering a methodology to define feasible architectural configurations in the face of constituent system failures. Ligaarden and Stølen [31] proposed sharing data among constituent systems as a form of heterogeneous redundancy, demonstrating its impact on increasing overall reliability. It is important to highlight that employing heterogeneous redundancy requires SoS reconfiguration to accommodate the new constituent systems.

Regarding reconfiguration in SoS, it has attracted attention within the research community, with different aspects being considered. For instance, Petitdemange *et al.* [16] introduced reconfiguration patterns to help reasoning on reconfiguration and maintaining the architectural patterns of an SoS. In a subsequent study [15], the authors proposed a design process for SoS reconfiguration. They recognize the need for SoS to adapt to environmental changes and undergo evolutionary development. The proposal was applied in the context of a realistic case study inspired in the French emergency services.

Wudka *et al.* [32] introduced an approach for decentralized SoS reconfiguration tailored to support open adaptive SoS. The authors presented the concept of strategy blueprints, which outline potential combinations of services provided by the constituent systems. During reconfiguration, each constituent system evaluates all strategies that can be instantiated given current conditions and selects the one that best fulfills predefined goals as the optimal target configuration.

Bhardwaj and Liggesmeyer [33] presented a proposal for a framework that facilitates the safe reconfiguration of an open adaptive system at runtime. Forte *et al.* [23] introduced a novel approach to reconfiguration within a smart product-based SoS. This approach, grounded in the SoS Engineering Lifecycle Concept, utilizes an IoT platform to ensure sustainability during SoS operation.

Regarding self-adaptation, Weyns and Anderson [34] presented three architectural styles for self-adaptation in SoS: Local Adaptations, Regional Monitoring–Local Adaptations, and Collaborative Adaptations. These styles vary in the degree of knowledge sharing and collaboration, addressing uncertainty at different scales. The Local Adaptations style prioritizes autonomy for constituent systems designers and compromises guarantees on cross-system uncertainties. In contrast, the Collaborative Adaptations style involves direct interaction between feedback loops, creating dependencies among constituent systems while enhancing support for cross-system property guarantees. The Regional Moni-

toring style strikes a balance by having feedback loops share information without direct interaction, offering a middle ground between the other two styles.

Wätzoldt *et al.* [35] developed a modelling language for collaborations in self-adaptive SoS that supports the explicit design of feedback loops. Maia *et al.* [36] explored the issue of defiant constituent systems, in which one or more constituent systems conflict with an SoS mission while pursuing their individual goals. Constituent systems that cannot be adapted to meet both individual and global requirements are referred to as "defiant". They proposed an approach termed "cautious adaptation" to dynamically adjust the behavior of constituent systems and satisfy the global mission when a conflict arises.

While these studies significantly contributed to the SoS body of knowledge, critical aspects remain unresolved. Firstly, existing literature on fault tolerance in SoS addresses representation aspects through modeling and specifying requirements for fault-tolerant SoS. In other words, they provide a comprehensive view from a fault tolerance perspective in SoS. Other studies delves into using heterogeneous redundancies to enhance fault tolerance, producing positive results. However, they fail to detail how disturbance detection would be carried out and how constituent systems are chosen as redundancies. This understanding is paramount for reconfigurations incorporating these constituent systems into the SoS.

Regarding studies concerning SoS reconfiguration, their focus has predominantly been on optimization, resilience, security, and sustainability of SoS. There is a need to explore how reconfigurations can be used for fault tolerance considering the dynamic nature of SoS, especially when leveraging heterogeneous redundancies. Moreover, no studies address the possibility that constituent systems might temporarily change their behavior, accidentally or intentionally. This aspect holds significant importance in the SoS context since they are primarily designed with "what is available" [37], aiming to fulfill missions rather than prioritizing optimization. Therefore, when an SoS undergoes reconfiguration using heterogeneous redundancies, the new architectural configuration may experience losses of security, performance, and privacy, among others. This is because constituent systems used as heterogeneous redundancies might not possess the same functionalities as the replaced systems. Consequently, the new architectural configuration often relaxes non-critical requirements to prioritize the critical ones. In addition, a new architectural configuration could also lead to additional costs from integrating new constituent systems. In this context, it is reasonable to assume that the originally designed architectural configuration is "the best fitted" for an SoS needs. Consequently, using the initially designed architectural configuration should be the preferred choice whenever possible - a factor that has not been considered in the SoS research to the best of our knowledge.

4. Research Method

The development of the proposed solution was guided by the principles of Design Science [38], a methodology commonly used in software engineering and information systems research for the design, evaluation, and refinement of software artifacts. As illustrated in Figure 1, our research method consists of four main steps: **Step 1** - Problem investigation, **Step 2** - Definition of the objective, **Step 3** - Solution design, and **Step 4** - Evaluation. The following subsections provide detailed descriptions of each step and their respective sub-steps, while Section 5 outlines the framework developed in Step 3, and Section 6 reports the results from Step 4.

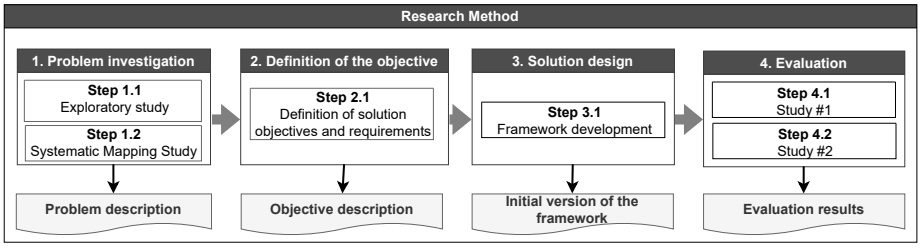

Figure 1: Research method.

4.1. Step 1 - Problem Investigation

This step consists of understanding and describing the problem to be addressed [38]. It involves identifying the needs and challenges that motivate the research. This process includes observing and analyzing the current context and identifying gaps or opportunities for improvement that justify the need to create a new solution or enhance an existing one [38]. To do so, we performed an exploratory study in a large Brazilian public organization [11], in which we observed the challenges regarding SoS reliability in the organizational context and how they are handled in practice.

In the exploratory study, we noticed situations in which the organization encountered difficulties arising from the dynamism of the SoS, specifically in managing transient failures and unreported implementation changes within constituent systems. To mitigate this, the organization implemented a fault tolerance strategy of activating alternative constituent systems to reduce the impact caused by the affected systems. This approach often resulted in a trade-off of quality attributes, as the SoS experienced a decrease in performance. Nevertheless, such performance loss was considered acceptable due to the inherent risks associated with the non-operation of an SoS.

Additionally, we carried out a Systematic Mapping Study (SMS) regarding SoS reliability [3], which investigated the state of the art of SoS reliability from three perspectives: factors that affect SoS reliability, approaches used to improve SoS reliability, and metrics used to assess SoS reliability. By analyzing

the 27 primary studies selected in the work, we observed a similar finding to the exploratory study: **heterogeneous redundancy has emerged as a strategy to enhance SoS reliability**. However, existing literature overlooks the dynamic nature when employing heterogeneous redundancies. Although this strategy can sometimes be a valuable solution when no other alternatives are available, it can lead to potential losses of performance, security, and privacy, among others. Additionally, such reconfigurations may increase costs due to integrating new systems. Given these factors, it is often most appropriate for SoS to operate in the original architectural configuration whenever possible. This perspective is overlooked in current literature.

4.2. Step 2 - Definition of Objective

Based on the results from Step 1, we defined that our solution should assist professionals in designing fault-tolerant SoS considering their dynamic nature. As a key requirement, we defined that our solution must be adaptable. This is because different SoS domains have unique requirements, nuances, and specific characteristics. In this context, adaptability ensures that the solution can be customized regardless of the domain.

Additionally, a requirement for the solution is to leverage the opportunistic nature of SoS design. In other words, the focus is on utilizing "at hand" systems as heterogeneous redundancies, taking advantage of the capabilities of these existing systems to fulfill the overall mission of the SoS [37].

4.3. Step 3 - Solution Design

We developed a prescriptive framework called ReViTA to meet the solution objectives defined in Step 2. **A prescriptive framework is a structured set of activities that provides specific directions on how to solve a problem**. It dictates which steps should be followed to ensure a goal's achievement [39]. Precisely, the framework's outcomes inform the monitoring and reconfiguration processes, which are crucial for enabling SoS fault tolerance by employing heterogeneous redundancies. In other words, our goal is not to introduce specific techniques for monitoring and reconfiguration, given the plethora of existing solutions for these purposes. Instead, we aim to assist professionals in implementing these techniques, guided by a systematic approach focused on fault tolerance that considers the dynamic and opportunistic nature of SoS.

To design our framework, we based on CM4SR, a conceptual model for SoS reliability [40]. Through a robust theoretical foundation, we developed CM4SR, encompassing 29 SoS reliability concepts and their relationships, fostering a comprehensive understanding of the subject. Section 5 details ReViTA and presents the CM4SR propositions that underlie its design.

4.4. Step 4 - Evaluation

In this research, our focus is on the framework's acceptance by professionals. Hence, we seek to understand how ReViTA is perceived and used by professionals to gain insights into its usefulness and ease of use. By gathering feedback

and insights from practitioners, we aim to investigate the feasibility of ReViTA as an novel artifact to improve SoS fault tolerance.

We carried out two studies involving professionals with different backgrounds. We conducted semi-structured interviews in both studies to gather the participants' feedback. In the first study (*Study #1*), the framework was introduced to the participants, followed by individual interviews to collect their perceptions of the framework from an intuitive perspective. In the second study (*Study #2*), the participants employed ReViTA based on a realistic case of SoS. Then, they exposed their opinions from a practical perspective.

In semi-structured interviews, a guide is developed with questions and topics that must be addressed [41]. The researcher has a clear objective but has little control over the discussion, i.e., questions are planned but are not necessarily asked in the same order as they are listed in the interview guide. For both studies, we defined primary questions that could be answered objectively using a Likert scale to capture a general perception from the participant about a specific aspect. Such closed-ended questions served as gateways to facilitate open-ended responses. Therefore, with each participant's response, we prompted follow-up with questions such as "*Why?*, "*Why do you think so?*", "*Can you elaborate on that?*" and others to continue gathering information as needed. This approach helps us to deep into the participants' thoughts, providing richer context and understanding beyond the initial response. Moreover, it encourages participants to reflect upon their answers, revealing additional insights [42].

Our interview guide included questions to evaluate the framework's usefulness and ease of use. By doing so, we address important factors that play a significant role in the acceptance of the framework. The Technology Acceptance Model (TAM) [43] inspired us to do so. TAM is a well-established information systems model that explains how users accept and use a particular technology. It declares that perceived usefulness and ease of use primarily influence technology acceptance. As Davis [43] explains, perceived usefulness is the degree to which an individual believes that using a particular technology will enhance their job performance. On the other hand, perceived ease of use is the degree to which an individual believes that using a particular technology will be free of effort. Even if an individual perceives a technology to be useful, its use could be hindered if perceived as too complex or requiring excessive effort.

We drew inspiration from TAM constructs to formulate our interview guide and perform a qualitative analysis. By doing so, we benefited from robust theoretical foundations enriched by years of research and empirical validation. This approach allowed us to capture the participants' perceptions while benefiting from TAM's theoretical solid grounding.

We recorded and transcribed the interviews for further analysis. We employed open coding procedures to support the analysis of participants' responses. This systematic approach allows us to analyze the responses and identify common themes and patterns. Open coding offers a structured way to understand the raw data, enabling us to categorize the responses based on their key elements, contributing to a proper data interpretation [44].

4.4.1. Scenario Description

An unique real-world scenario was used as reference for both studies. In *Study #1*, the scenario helps us demonstrate the application of ReViTA. Participants used the same scenario as a reference to apply the framework in *Study #2*. Such a scenario is known by all participants of both studies. This helped the participants better understand and use the framework. **The participants in both studies were distinct** to ensure that participants in *Study #2* were not influenced by any information received in *Study #1*.

The scenario refers to an SoS that supports the responses to power outage incidents at a large Brazilian public university with distributed institutes and campuses. An interruption in the power supply would result in significant loss to the university, such as the unavailability of the IT services that support the daily operation of the university. The importance of this SoS was particularly underscored during the COVID-19 pandemic, as the university staff had to work remotely as part of efforts to minimize the spread of the coronavirus[1]. Since no employee was on site to notice any power outage, the detection and notification of such events had to be performed automatically. Therefore, it is a real-world scenario requiring fault tolerance, making it a practical and meaningful context for evaluating ReViTA.

The SoS under consideration is a directed SoS composed of four constituent systems: Power Generator Management System, General Power System, Uninterruptible Power Supply Management System, and Telephony Gateway. Different departments independently maintain the constituent systems. The constituent systems of this SoS are described in Table 1.

Constituent system	Description
Power Generator Management System (PGMS)	A system maintained and operated by the campus electrical department. It identifies local power outages and provides information on the fuel level in the generators.
General Power System (PSMS)	A system maintained and operated by the campus electrical department. It reports the status of the electricity supply at the main campus substation.
Uninterruptible Power Supply Management System (UPSMS)	A system maintained and operated by the campus electrical department. It provides information about the electrical energy consumption and the Uninterruptible Power Supply batteries' level of autonomy.
Telephony Gateway	A system maintained and operated by the IT department. It notifies the person in charge of an interruption in the power supply via a telephone call.

Table 1: Constituent systems of the SoS that support response to power outage incidents.

In this SoS, an additional system was developed to mediate communication among constituent systems. We call this system as the Orchestrator. The SoS operates as follows:

1. The Orchestrator monitors the status of the local electricity supply through periodical queries to the PGMS;

[1] `https://www.who.int/westernpacific/emergencies/covid-19/information/physical-distancing`

2. If an interruption is detected, the Orchestrator waits 300 seconds and makes a new query. Sometimes the supply is interrupted for a few seconds, and this 300-second wait prevents the on-call professional from being called unnecessarily on weekends or at dawn;
3. If the power supply remains interrupted after 300 seconds, the Orchestrator queries the fuel level in the PGMS. In addition, it consults the UPSMS to collect information on current power consumption and the Uninterruptible Power Supply autonomy (determined by battery charge);
4. The Orchestrator calculates the total autonomy, considering the current power consumption, the fuel level of the generators, and the autonomy level of the Uninterruptible Power Supply batteries;
5. The Orchestrator consults the PSMS to verify whether the interruption is local or campus-wide;
6. The Telephony Gateway sends a voice recording to the on-call phone, informing about the power outage and the total autonomy time.

The constituent systems are independent and serve specific purposes. Moreover, the implementation of the SoS was an initiative of the IT department, which requested access to the constituent systems maintained by the electrical engineering department (PGMS, PSMS, UPSMS). However, the electrical engineering department is not engaged in the operation of the SoS. This lack of engagement implies problems that compromise the response to the interruption in the electricity supply on the campus. The main problems are:

- **P1 - Telephony Gateway unavailability**: this constituent system is installed far from the data center and lacks the appropriate infrastructure. The network switch that provides connectivity to the Telephony Gateway is connected to an old Uninterruptible Power Supply with little autonomy. Sometimes, during an interruption in the power supply, the Uninterruptible Power Supply shuts down within a few seconds, causing the network connectivity devices to shut down. When this occurs, the Orchestrator cannot activate the Telephony Gateway to make the phone call;

- **P2 - UPSMS unavailability**: this constituent system experiences the same problem as the Telephony Gateway. When it becomes unavailable, it is not possible to verify whether the outage is local or campus-wide;

- **P3 - Fuel Level Sensor Unavailability**: the fuel sensors of PGMS stop working due to unknown reasons. When this occurs, it is not possible to calculate the total autonomy. It is necessary to manually reset the sensor for it to return to operation.

4.4.2. Study #1

In this study, we focused on understanding the participants' perception of the framework's clarity and understanding, ease of use, usefulness, potential to reduce effort, completeness, intention of use, and adaptability. These criteria

help assess the framework's feasibility for the intended users and their context. By gathering opinions from professionals, the evaluation can capture their subjective experiences and perspectives, highlighting strengths and areas for improvement. This feedback is crucial in guiding enhancements and adjustments to increase its chances of successful adoption.

For this study, we defined the following research question: **RQ1** - *How is the acceptance of ReViTA among professionals? (RQ1)*

The study's dynamic involved introducing the framework to the participant and detailing all activities, including their inputs and outputs. Then, we demonstrated the framework's use in the scenario described in Section 4.4.1[2]. Following that, we carried out the interviews.

The participants of *Study #1* (14) were selected because of their experience deploying an SoS that is in active use within large Brazilian public university. This ensures that the participants have some familiarity with the complexities of designing and implementing an SoS, making their feedback and insights all the more valuable to our study.

In summary, all participants have an academic background in computer science. Most participants (8 out of 14) hold an M.Sc., three have a B.Sc., and three have a Ph.D. Ten participants have more than ten years of professional experience, and four have between five and ten years of experience. Eight participants work as software developers, with three of these also working as software architects. Four participants are support analysts, and two are IT managers. Table 2 presents the participants' profiles.

We presented seven statements to the participants and asked them to express their opinion on specific aspects of our framework using a 5-point Likert scale (*Strongly Agree, Partially Agree, Neutral, Partially Disagree, and Strongly Disagree*). The statements and related aspects are listed in Table 3. The statements from S1 to S6 are derived from TAM's aspects. Additionally, we included a statement (S7) to evaluate ReViTA's adherence to the requirement of adaptability defined in Step 2. We encouraged participants to provide detailed insights and discuss their perceptions of each aspect. Before proceeding with the interviews, we conducted a pilot with a software developer to identify potential improvements in the study's design. After adjusting our interview guide, we carried out the interviews between October 14th and 24th, 2022. Each interview session lasted approximately one hour.

4.4.3. Study #2

This study aimed to evaluate ReViTA's acceptance from a practical perspective. We revisited some aspects addressed in *Study #1* and introduced new aspects that only those who employed the framework could evaluate. Specifically, our focus was to gather information from participants about their learning experiences while using the framework, their perception of clarity and under-

[2]A document with the demonstration of ReViTA, as presented to the participants, is available in English at the following link: `https://doi.org/10.5281/zenodo.8102848`

ID	Acad. degree	Exp. (years)	Current position	Software development	Software architecture	Software reliability	IT mgmt.
P1	M.Sc.	15-20	IT manager	Very high	Average	Low	Very high
P2	M.Sc.	5-10	Support analyst	Very high	High	Low	High
P3	M.Sc.	5-10	Support analyst	High	Average	Average	Very high
P4	B.Sc.	20+	Support analyst	Very high	Low	Low	High
P5	B.Sc.	15-20	Support analyst	High	High	Average	Very low
P6	M.Sc.	10-15	Software developer and Software architect	High	High	No experience	High
P7	Ph.D.	5-10	Software developer	Very high	Average	Low	Average
P8	M.Sc.	15-20	IT manager	Very high	Very high	Very high	High
P9	B.Sc.	15-20	Software developer	Very high	Low	No experience	Low
P10	M.Sc.	5-10	Software developer and Software architect	Very high	Very high	High	Average
P11	Ph.D.	15-20	Software developer	Very high	Low	Average	Average
P12	M.Sc.	20+	Software developer	Very high	High	Low	Low
P13	M.Sc.	10-15	Software developer	Very high	High	High	Very low
P14	Ph.D.	15-20	Software developer and Software architect	Very high	High	High	Low

Table 2: Study #1 - Participants' profile.

standing, ease of use, usefulness, and impact on job performance. For this study, we defined the following question: **RQ2** - *Is ReViTA feasible from a practical perspective?*

Four professionals, distinct from *Study #1*, participated in this study, as in Table 4. All participants support analysts (i.e., they work on the maintenance and troubleshooting of software-intensive systems, including the SoS in operation at the university) with more than 5 years of experience. Three of them hold an M.Sc. and one holds a B.Sc. We selected them due to their expertise in the SoS operation. They have a daily routine of troubleshooting, and operating the SoS. Therefore, participants have a deeper and more practical understanding of the problems and challenges faced in the context of the SoS, allowing for a richer and more detailed analysis of the results.

For this study, we developed a tool that provides a simplified way for performing the framework activities, mainly through diagrammatic notations and textual descriptions, without relying on highly specialized approaches. We made

ID	Statement	Aspect	Description
S1	The framework and its activities are easy to understand.	Clarity and understanding	It refers to how easily a participant can understand the framework and how clear its activities and related outputs are to them.
S2	The framework activities are easy to perform.	Ease of use	It refers to the degree to which a participant believes that using the framework would be free from difficulty.
S3	The framework is useful.	Usefulness	It refers to the degree to which a participant believes that using the framework would produce desirable outcomes.
S4	Using this framework would reduce the effort to design fault-tolerant SoS.	Potential to reduce effort	It refers to the perception of how much the framework can decrease the amount of work or effort required to accomplish a goal.
S5	All framework activities are necessary and enough.	Completeness	It refers to having all necessary activities needed to achieve a goal.
S6	I will use this framework if I have the opportunity.	Intention of use	It refers to the likelihood that a participant plans to use the framework in the future.
S7	The framework is adaptable to all SoS contexts	Adaptability	It refers to the degree to which the framework can be adjusted to different SoS contexts.

Table 3: Study #1 Statements.

ID	Academic degree	Experience	Current position
P15	M.Sc.	5 - 10 years	Support analyst
P16	M.Sc.	5 - 10 years	Support analyst
P17	B.Sc.	5 - 10 years	Support analyst
P18	M.Sc.	10 - 15 years	Support analyst

Table 4: Study #2 - Participants' profile.

this decision because implementing specialized approaches would require a level of knowledge and experience that the participants currently do not possess. By doing so, we ensured that participants could effectively engage with the framework's concepts and activities without being hindered by a lack of prior knowledge. Moreover, this approach allows the participants to focus on understanding the fundamentals of the framework and its potential benefits in improving SoS fault tolerance, setting the basis for improvements.

The dynamics of this study involved training on the framework and the tool. After the training, the participants performed the framework activities having the SoS described in Section 4.4.1 as a reference. Participants employed the tool to perform the activities of the framework ReViTA, considering their experience with the SoS described and their knowledge about it.

We formulated five statements for *Study #2*, using a 5-point Likert scale (*Strongly Agree, Partially Agree, Neutral, Partially Disagree, and Strongly Disagree*) to gather the first impression of the participants. Then, we encouraged the participants to deep into their answers by asking follow-up questions to gain a more comprehensive understanding. Table 5 lists the statements of *Study #2*.

Before proceeding with this study, we carried out a pilot with a software engineer, which led us to make simple adjustments in the interview guide. The interviews occurred on May 5th and 6th, 2023.

ID	Statement	Aspect	Description
S8	I found the process of learning to use the framework to be a positive experience.	Learning experience	It refers to the participants' process of gaining knowledge related to the use of the framework and how intuitive the framework is perceived.
S9	The framework's purposes and activities were clear and understandable.	Clarity and understanding	It refers to how easily a participant can understand the framework and how clear its activities and related outputs are to them.
S10	I found the framework to be easy to use.	Ease of use	It refers to the degree to which a participant believes that using the framework would be free from difficulty.
S11	This framework would be useful in supporting me to do my job.	Usefulness	It refers to the degree to which a participant believes that using the framework would produce desirable outcomes.
S12	Applying this framework would allow me to perform my job more efficiently.	Impact on job performance	It refers to the degree to which the participants believes that using the framework would improve job efficiency, or allow them to accomplish tasks more quickly.

Table 5: Study #2 statements.

5. The ReViTA Framework

ReViTA is a prescriptive framework comprising activities aimed at supporting the design process of fault tolerance oriented reconfigurations in SoS. ReViTA supports designing fault-tolerant reconfigurations by prescribing activities that inform the **monitoring** and **reconfiguration** processes. As Petitdemange *et al.* [15] explain, the lifecycle of SoS reconfiguration involves continuous SoS monitoring. When a monitor detects a situation in which the operational architectural configuration no longer meets the reliability requirements, it triggers a reconfiguration process. This reconfiguration process presumes that the target architectural configuration is already known.

Such a monitoring process requires continuous and precise observation of the behavioral attributes of the constituent systems. Hence, it is imperative to understand the conditions in which they must operate to contribute to an SoS. This understanding is critical to ensuring that the monitoring process accurately identifies situations in which the architectural configuration no longer meets the reliability requirements.

Additionally, ReViTa encompasses activities for supporting the design of architectural configurations with heterogeneous redundancies. These configurations inform the reconfiguration process. By following these activities, professionals can leverage the opportunistic nature of SoS to design architectural configurations that maintain SoS ability to achieve its core mission, thus increasing SoS reliability.

As a prescriptive framework, ReViTA can be applied with different tools, methods, and processes, allowing for customization to better meet the requirements of each SoS. This approach aims to achieve the fulfillment of the adaptability requirement outlined in Section 4.2.

5.1. Overview

To design ReViTA, we introduced the concept of **Transient Architectural Configurations** (TAC), which are alternative architectural configurations (i.e., configurations with a different composition of constituent systems from the desirable architectural configuration) that operate when the desirable architectural configuration experiences disturbances resulting from failures or undesired behavior of one or more constituent systems. We named it "transient" because we assume the desirable architectural configuration should operate whenever possible, as it best satisfies all predefined requirements for the SoS, critical or non-critical. Therefore, our framework envisages continuous monitoring of constituent systems so that the desired architectural configuration can resume operation as soon as the disturbances cease.

Figure 2 illustrates the dynamics of SoS using TAC. *TAC X* and *TAC Y* are alternative architectural configurations that differ from the desired SoS configuration. These configurations involve different constituent systems but serve the same mission, albeit with some loss in performance or capabilities. The desired architectural configuration is operational at time *t1*. A monitor detects a disturbance at time *t2*, and the SoS is reconfigured to *TAC X*. At time *t3*, the monitor determines that the previously identified disturbance no longer exists, and the SoS returns to the desired architectural configuration. At time *t4*, a new disturbance, distinct from the one at time *t2*, is identified, leading the SoS to be reconfigured to *TAC Y*, which specifically addresses this new disturbance. At time *t5*, the monitor recognizes that the disturbance ceased, and the SoS reverts to the desired architectural configuration.

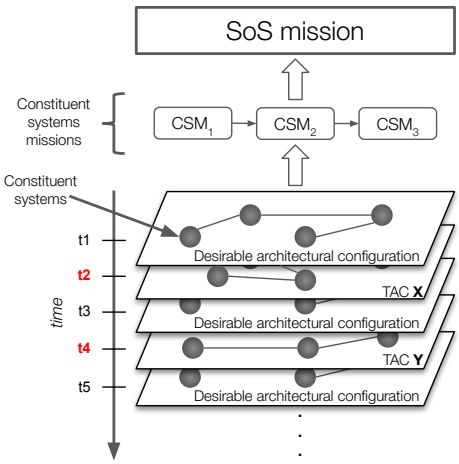

Figure 2: SoS dynamics over time using TAC.

The input of the framework is a description of SoS goals and the outcomes are the information required for the monitoring and reconfiguration processes to enable fault tolerance through reconfigurations. Figure 3 shows that ReViTA has two major steps and two cycles. The smaller cycle represents the continuous

monitoring activity essential for timely reconfigurations. By constantly monitoring the state of SoS, potential disturbances can be detected in time, enabling timely reconfigurations to mitigate or prevent disturbances at the SoS-level. The larger cycle indicates that ReViTA should be applied continuously as the missions of an SoS evolve, and the fault tolerance mechanisms need to keep up with this evolution.

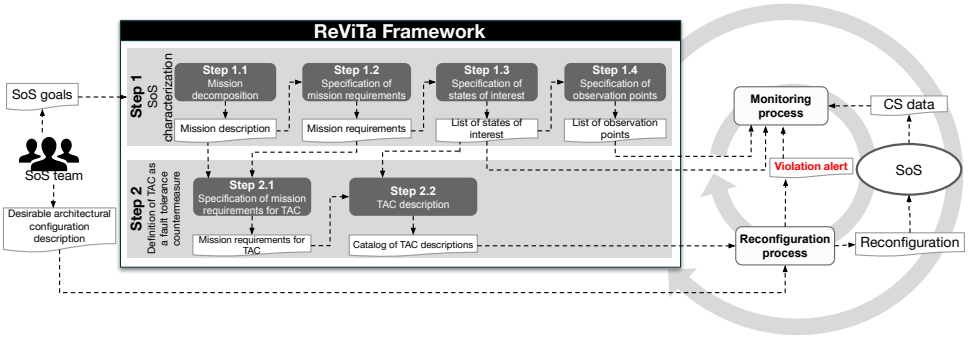

Figure 3: ReViTA overview.

The two major steps are discussed individually in the following sections. These are the "**SoS Characterization**" and the "**Definition of TAC as a Fault Tolerance Countermeasure**".

5.2. SoS Characterization

SoS Characterization step provides a structural and functional view of the SoS and its constituent systems. It aims to identify constituent systems and their desired behavior while fulfilling its missions. In addition, it identifies interfaces to verify the behavior of the constituent systems. This step has four activities: *Mission Decomposition, Specification of Mission Requirements, Specification of States of Interest,* and *Specification of Observation Points.* The outcomes of this step offer a comprehensive understanding of the SoS structure and the behavioral aspects of the constituent systems, which are required by the monitoring process.

5.2.1. Mission Decomposition

As highlighted in CM4SR [40], SoS reliability is closely related to its ability to fulfill its missions successfully. These missions are made possible through the interaction among the constituent systems, each fulfilling their individual missions. Consequently, to ensure fault tolerance in an SoS, and ultimately to maintain its reliability, we need a comprehensive understanding of the responsibilities of each constituent system, and their interactions within an SoS. This understanding is crucial as each constituent system is a potential point of disturbance that can severely impact the overall SoS reliability. Achieving

such a comprehensive understanding can be reached through a detailed mission decomposition process.

Mission Decomposition is intended to provide a functional coarse grain view of the SoS overall mission [45]. It involves breaking down the main mission into smaller missions, such as SoS sub-missions and constituent systems missions. By doing so, it is possible to identify the specific responsibilities of each constituent system and the relationships between them. This prescriptive activity provides a comprehensive understanding of SoS goals and responsibilities, and it helps to identify potential sources of disturbances.

The outcome of "*Mission Decomposition*" is a high-level description of missions, sub-missions, and individual missions. In addition, the mission decomposition provides a description of the relationship among missions and informs the responsibilities of constituent systems. Different tools can be used to represent mission decomposition, such as KAOS [46], mKAOS [47, 11], and SysML [48, 45]. Figure 4 illustrates the result of the mission decomposition of the SoS that supports the response to power outage incidents on the campus of a large Brazilian public university, described in Section 4, using the mKAOS notation [47]. The blue rectangle at the tree's root represents an SoS main mission, which undergoes refinement (represented by the yellow circle) into sub-missions until reaching the level of individual missions. The constituent systems, represented by orange diamonds, fulfill these individual missions.

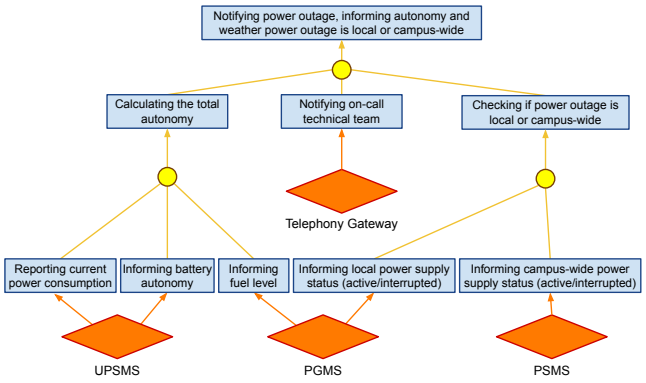

Figure 4: Mission decomposition with mKAOS.

mKAOS is a mission-oriented language created specifically to represent information associated with an SoS mission. The mKAOS language follows the conceptual model of SoS missions, proposed by Silva *et al.* [49], who identified elements that must be considered in the SoS context.

5.2.2. Specification of Mission Requirements

This activity complements "*Mission Decomposition*" by specifying information on constituent systems' behavioral aspects. Here, we identify the requirements that constituent systems must meet to ensure that their individual mis-

sions align with the expectations of an SoS. Essentially, mission requirements determine how constituent systems must provide individual capabilities to SoS. As explained in CM4SR, this is particularly important for reliability, as an SoS can only effectively fulfill its missions if the capabilities at the SoS level are adequately provided. These SoS-level capabilities, in turn, depend on the constituent systems properly accomplishing their individual missions. To do so, the individual capabilities of the constituent systems must be properly provided [40].

For example, if a constituent system's mission involves data processing, this constituent system must meet requirements concerning the accuracy, speed, and reliability of data processing. Failure to meet these requirements could lead to disturbances such as data errors or delays, causing a cascading effect throughout the SoS and compromising the efficiency and effectiveness of other constituent systems that rely on this processed data.

Therefore, in addition to mission decomposition, a comprehensive assessment of the requirements for each constituent system is critical in ensuring SoS fault tolerance. The outcome of this activity is the specification of the requirements that each constituent system must meet to provide capabilities and accomplish individual missions. It details conditions, such as time, quantity, and standards, among others.

5.2.3. Specification of States of Interest

In general, this activity seeks to inform the conditions under which a constituent system can meet its mission requirements. The ability of a constituent system to fulfill its mission requirements is directly related to its observable behavior [8]. In other words, if the constituent system starts behaving differently, it may not be able to meet its mission requirements. A constituent system can change its behavior for various reasons. These may include planned (but sometimes not informed) evolution to meet individual requirements or respond to other stimuli, such as attacks, overload, or hardware issues.

The change of behavior of a constituent system, in turn, produces symptoms revealed by several measurable attributes, such as availability, response time, memory utilization, and processor load, to mention a few. The measures of these attributes characterize the operational state of the constituent system. Consequently, it is possible to verify the behavior of the constituent systems by observing their operational states through measurable attributes [40]. Hence, when a constituent system exhibits behavior in alignment with the needs of an SoS by fulfilling its mission requirements, it is operating in a state of interest. Essentially, it behaves desirably in the context of SoS.

For example, Equation 1 illustrates the representation of the state of a constituent system through two properties: processor load (*proc_load*) and response time (*resp_time*). The constituent system β operates in the state of interest (value 1) if the processor load is less than 90% and the response time is less than 10 milliseconds. Otherwise, the constituent system β cannot contribute to SoS (value 0), which is considered a violation of the state of interest.

$$cs\_state_{\beta} = \begin{cases} \mathbf{1}, & \text{if } proc\_load < 90\% \text{ and } resp\_time < 10ms \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad (1)$$

The outcome of this activity is a description of the measurable attributes that characterize the state of interest of each constituent system. This outcome informs the monitoring process, which uses this data to trigger SoS reconfiguration. The properties that characterize the state of interest of the constituent system must be carefully defined, as it requires a precise selection of properties and accurate calibration of the measures.

5.2.4. Specification of Observation Points

A monitor is designed to continually observe and evaluate the state of the constituent systems, which enables immediate detection of undesirable behaviors. This detection is vital because the sooner a problem is detected, the sooner a response can be initiated, reducing the potential negative impact. Therefore, this activity comprises the specification of observation points, which are technical means to continuously observe the states of constituent systems [50], as illustrated in Figure 5.

Observation points expose data from behavioral attributes of the constituent systems, such as hardware, network, operating system, and end-user applications [50]. Table 6 lists examples of potential observation points that can be used to monitor constituent systems, according to Lampesberger *et al.* [50]. The number and types of observation points and data types depend on the level of control over the constituent systems.

Category	**Description**
Hardware	System and processor temperatures, voltages, fan speeds, memory failure counters, hard drive health, and performance counters.
Network	Device availability, error rate, throughput, and response time.
Operating system	Logs for crashes, debug information, notifications, and events in general. Also, an operating system typically collects runtime performance metrics such as system load, processor load, memory utilization, and network interface utilization.
Service	Services may maintain individual logs for various purposes, for example, service-specific events, performance counters, transaction logs for database systems, and access logs for auditing tasks, to name a few.
Middleware	Message routing, coordinated actions, and service orchestration, to name a few. Moreover, the logging capabilities in middleware components, typically for debugging and auditing, can be valuable observation points.
User	The platform or software provided may provide user-centric logging for auditing or service adaptation, e.g., history, access, authentication, or geolocation logs.

Table 6: List of potential observation points [50].

Observation points can be classified as *white-box* or *black box*. In the case of *white-box* observation points, it is assumed that the points can be instrumented according to the needs of SoS, granting direct access to the constituent systems. Alternatively, *black-box* observation points operate under the assumption that there is no direct access to the constituent system, and the system's state is inferred by monitoring the available interfaces. For instance, in systems connected to TCP/IP networks, one can estimate processor load by monitoring response times through the ICMP [51].

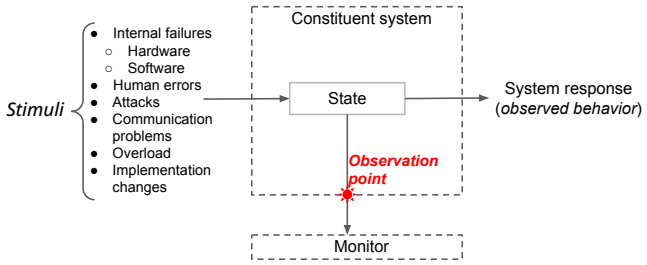

Figure 5: Dynamics of observation points.

The more accurately the observation points reflect the state of the systems, the more precise the runtime monitoring and subsequent actions will be. Identifying observation points may require a comprehensive understanding of each constituent system within the SoS. This includes understanding the constituent system's functionality, interfaces, and data formats. In particular, understanding what data is produced, where it is stored, and how it can be accessed is fundamental in specifying observation points.

The outcome of this activity, which comprises a specification of the observation points and associated properties of the constituent systems, is essential to the monitoring process. This outcome also details operational information about data collection procedures and data formats.

5.3. Definition of TAC as a Fault Tolerance Countermeasure

TAC serves as a fault tolerance countermeasure designed to respond to the disturbances resulting from the undesirable behavior of constituent systems. TAC Description requires a deep understanding of SoS structure and potential points of disturbance, which is why it is tied to the outcomes of SoS Characterization. This step consists of two main activities: "*Specification of Mission Requirements for TAC*" and "*TAC Description*".

5.3.1. Specification of Mission Requirements for TAC

This activity consists of analyzing the mission requirements, prioritizing the critical ones, and relaxing the non-critical ones. Relaxing mission requirements is an essential strategy to ensure fault tolerance in SoS. A certain degree of relaxation of mission requirements may be necessary to allow an SoS to fulfill an adapted version of the original mission. It can help an SoS adapt to changes in operational conditions and ensure it meets the mission's critical requirements. Relaxing mission requirements is a key strategy for addressing the severity of disturbances as it introduces a degree of versatility to SoS, enabling it to adopt various TAC. This flexibility expands the possibility of using constituent systems that only partially align with the initial mission requirements. Still, they can contribute in the face of disturbances.

The activity aligns with the opportunistic nature of SoS design, emphasizing achieving the main mission rather than optimizing overall performance. In other

words, the focus is not on achieving optimization but on realizing functional success amid real-world constraints. Relaxing mission requirements amplifies the likelihood of mission success.

Professionals can design different TAC to harness available constituent systems by relaxing requirements. While possibly not meeting primary requirements fully, these constituent systems can still contribute to SoS when the primary constituent systems are producing disturbances. This promotes an adaptable design, maximizing the use of existing resources and increasing the mission's success chances.

Furthermore, this activity acknowledges the inherent complexity and unpredictability of SoS environments. Relaxation is a proactive measure against the problems caused by an SoS dynamic nature. In SoS, disturbances are often unavoidable, and by infusing adaptability into mission requirements, we bolster the SoS reliability.

5.4. TAC Description

This activity encompasses the design of the TAC following the mission requirements specified in the previous activity. The focus of this research is not to propose a method for TAC design, as it is a complex task that is affected by many contextual aspects. Moreover, there are already numerous approaches to this objective. Each of these methods considers specific aspects, such as testability [37], security [52] and costs [53], among others. This activity results in a set of architectural descriptions of TAC that feeds the reconfiguration process. These descriptions provide detailed information regarding the constituent systems and their interactions within the SoS. Furthermore, they can encapsulate various elements, such as interaction interfaces, system behaviors, and constraints.

Each TAC must be linked to one or more specific state of interest violations. To manage situations where several TACs relate to a single violation, assigning a priority to each TAC is vital. This allows the reconfiguration agent to select and implement the most suitable TAC given the occurrence of a particular disturbance. Consequently, a TAC can be succinctly represented as a 3-tuple, detailed as follows:

$$TAC_X = \{architectural\_description_X, violation_X, priority_X\} \tag{2}$$

TAC can be described using Architectural Description Languages (ADLs). They can be used to facilitate system design communication by providing means to describe system structures and behavior. An ADL can be formal, semi-formal, or informal. Formal ADLs support formal verification and analysis, meaning they can automatically detect potential design flaws or verify the conformity of a system to its specifications. However, their usage requires a deep understanding of mathematical concepts and logic, which might be complex and time-consuming. Semi-formal ADLs offer a balance between precision and usability. They use graphical notations, similar to UML, which are easier to understand than formal languages. They provide some level of formality, i.e.,

they define precise syntax and somewhat precise semantics, but they typically do not support the same depth of automatic analysis as formal languages. Informal ADLs are mostly textual and graphical, with loose or no semantics, and are mainly used for communication and documentation purposes. They are easy to use but lack the precision for automatic analysis or verification. Examples include block diagrams or simple component diagrams.

For the sake of simplicity in our evaluation studies, we chose to utilize the mKAOS language to describe TACs. Originally, mKAOS was designed to specify SoS missions [47]. However, it also conveniently outlines some architectural elements, such as constituent systems, and describes how they interact to achieve those missions. The simplicity of use and the ease of comprehension make mKAOS an ideal language for the acceptance evaluation of ReViTA. Using mKAOS helps simplify the process, making it easier for the participants to understand and assess the framework.

Figure 6 shows an example of a TAC description using mKAOS. Such TAC is related to the problems of unavailability of the telephony gateway (P1), unavailability of UPMS (P2), and unavailability of the fuel sensors (P3), described in Section 3.4. In this case, the TAC relaxes the autonomy calculation requirement, removing it as it is not possible to calculate it due to the unavailability of information provided by the UPSM (battery autonomy) and the sensors (fuel level). Furthermore, the TAC includes an email server replacing the telephony gateway, ensuring that the on-call technical team is alerted via email. Note that details regarding architectural aspects, such as communication interface definitions, are omitted, with the focus solely on the composition of the SoS.

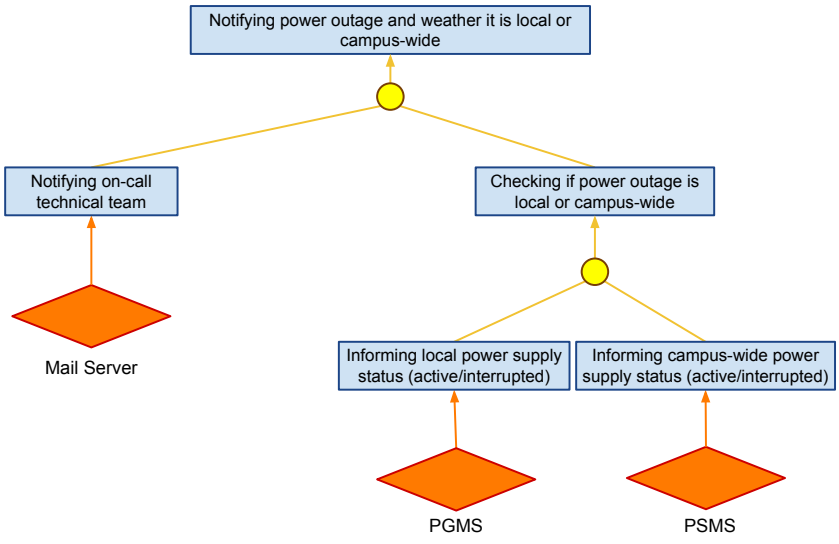

Figure 6: TAC associated with P1, P2, and P3 problems.

6. Results

The results from the two evaluation studies are presented in this section.

6.1. Study #1

We conducted this study with 14 participants and introduced them to the framework. We then asked for their opinions. Figure 7 shows a summary of the responses from participants about the statements we provided in Table 3. In short, participants generally had a positive perception of ReViTA. The upcoming sections contain detailed information about the participants' responses.

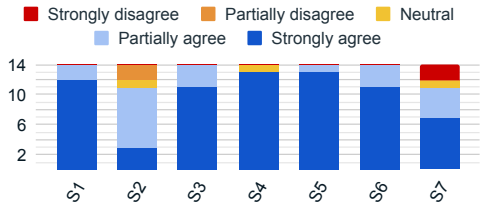

Figure 7: *Study #1* - Participants' responses.

6.1.1. Clarity and Understanding (S1)

The majority of participants (12 out of 14) demonstrated consensus regarding the clarity and understanding of the prescriptive activities contained in the framework. They attributed this perception to the sequential logic of the activities, in which the output of one activity seamlessly informs the next, contributing to an overall coherent framework. P10 stated: "*I liked that the framework starts from a good assumption, which is the focus on the mission. And all other activities are guided by the result of the mission decomposition. This structure of the framework was very clever*".

Additionally, two participants highlighted the value of having examples of familiar contexts, indicating that real-world applications enhance the participants' comprehension of the framework. This suggests that presenting the framework with more practical illustrations could be a beneficial strategy for improving the framework's understanding.

P7 argued that ReViTA is "*somewhat abstract*". This observation points to an opportunity for enhancing the connection between the prescriptive activities of the framework and their practical implementation. Addressing this perception of abstraction may require additional efforts to ground the framework in concrete applications, fostering a more tangible understanding.

6.1.2. Ease of Use (S2)

The majority of participants (8 out of 14) expressed a partial agreement on ReViTA's ease of use, indicating a moderate level of ease in performing the

activities. A significant insight emerged during the discussion of the SoS Characterization step, specifically in the "*Mission Decomposition*" and "*Specification of Mission Requirements*" activities. Participants acknowledged the potential complexity and challenges inherent in dealing with large-scale systems.

Moreover, a key point of attention raised by participants regards the "*Specification of mission requirements for TAC*" activity. The participants highlighted the contextual nature of decisions regarding requirement relaxation, noting that it is rarely universally prescribed. This insight underscores the importance of a nuanced understanding of each SoS domain's characteristics, constraints, and objectives to use ReViTa properly. The participants also emphasized that decisions regarding requirement relaxation should be careful, strategic, and adapted to the circumstances. They recommended the involvement of domain experts when using ReViTA, emphasizing that more than just knowledge in information technologies may be required for the proper execution of activities and navigating the complexities inherent to different domains. This underscores the need for a nuanced and context-sensitive approach to employing ReViTA in diverse SoS scenarios.

6.1.3. Usefulness (S3)

11 out of 14 participants strongly agreed on the usefulness of ReViTA. They recognized the framework's capability to introduce "*a structured approach to problem domains traditionally addressed through intuitive means, relying on the experiential knowledge of professionals*". In this regard, the framework's guidance on designing fault tolerance-oriented reconfigurations was perceived as positive for its role in addressing the complexities that may arise in such systems. Participants appreciated the comprehensive view ReViTA offers, recognizing its potential to support decision-making processes when addressing fault tolerance. An example cited by the participants was the "*TAC Description*" activity, in which potential additional costs, such as acquiring constituent systems, could be incurred.

Moreover, participants acknowledged that ReViTA is a valuable artifact in determining the most suitable actions to take under specific circumstances. In this regard, P1 stated: "*by following this approach, we could make early decisions more assertively, I think. Because TACs are designed based on a structured approach, which considers what is critical and what are the potential forms of treatment for each known event that could result in failures*".

Participants also argued that applying ReViTA may be time-consuming, even for very simple scenarios. Their perception is that, in these circumstances, ReViTA could not be useful. For instance, P6 argued that "*performing the framework activities is not trivial, and it can pose unnecessary effort if an SoS operates in non-critical domains*". Despite this, they underscored the value of ReViTA in the context of large-scale and critical SoS, suggesting that its application may be most impactful in specific contexts characterized by scale and criticality.

6.1.4. Potential to Reduce Effort (S4)

The majority of participants (13 out of 14) strongly agreed that ReViTA has the potential to significantly reduce the effort required to address SoS fault tolerance. The main argument is that using ReViTA can produce comprehensive documentation and facilitate subsequent actions.

Furthermore, participants emphasized that the framework introduces a systematic approach to address critical aspects that might be ignored when relying solely on intuitive practices. For example, P1 highlighted the importance of mission decomposition activity, recognizing it as a critical step in ensuring fault tolerance by enabling the precise identification of system failure points. Traditionally, this process relies on individual experiences with failures, potentially ignoring a broader range of possible failures that could be easily identified through a systematic application, as the framework suggests. P5 reinforced this opinion by asserting that the effort required to handle fault tolerance becomes more qualitative, implying a shift toward a more comprehensive and systematic approach that adds value beyond mere quantitative measures.

6.1.5. Completeness (S5)

Most participants (13 out of 14) strongly agreed that completing all activities is essential for achieving the framework's objectives. The central rationale behind this perception stems from the interdependence among the activities. Participants emphasized that each activity plays a crucial role, and removing any individual activity would disrupt the seamless execution of the remaining tasks. This collective viewpoint highlights the interconnected nature of the activities and emphasizes their mutual dependence on the successful realization of the framework's intended outcomes. Only one participant partially agreed that all activities are indispensable. This participant attributed the response to a "*lack of comprehensive knowledge*" regarding the subject.

6.1.6. Intention of Use (S6)

11 participants strongly agreed that they could use ReViTA in future opportunities. They highlighted that ReViTA incorporates activities commonly employed in practice, activities that, while recognized, are often underutilized due to a lack of accurate information and appropriate support. This suggests that ReViTA has the potential to bridge the gap between theoretical knowledge and practical application.

P10, in particular, argued that ReViTA builds upon a robust foundation by emphasizing a clear understanding of overall goals, ultimately leading to the achievement of multiple TAC for specific problems: "*I really liked how it was designed, from the objective to how you will flex your architectural configurations to meet these objectives, you know? You end up establishing a prevention system*". Overall, the participants considered this objective-oriented approach highly valuable.

P11 further underscored the significance of adopting a structured approach to address the issue of fault tolerance in SoS. Recognizing the challenges in finding suitable approaches in existing literature for these intricate systems, P11

highlighted ReViTA's potential in filling this gap: "*I like to have a methodological approach to solving problems. We often do not find an appropriate approach in literature*".

Participants also emphasized the need for further refinement and customization to suit specific needs. This perception reflects a practical understanding that frameworks are most effective when tailored to the unique contexts and challenges of individual situations. However, even with this call for refinement, participants expressed confidence in the ReViTA's ability to provide an interesting approach to addressing fault tolerance challenges in SoS.

6.1.7. Adaptability (S7)

Half of the participants (7 out of 14) strongly agreed that the ReViTA framework is adaptable across all SoS contexts. Meanwhile, a smaller group of four participants expressed partial agreement. One maintained a neutral stance among the remaining participants, while two strongly disagreed.

Participants recognized the potential adaptability of ReViTA's generic activities and acknowledged the framework's ability to address common challenges in managing fault tolerance in SoS. In addition, they highlighted the framework's potential to provide systematic guidance and documentation, which can contribute to a more comprehensive and structured approach to handling fault tolerance.

However, there was a certain skepticism among some participants regarding the universal applicability of ReViTA in all SoS contexts. This perception arose from recognizing that SoS can exhibit unique characteristics, complexities, and uncertainties that a generic framework may not fully capture. In this regard, P11 stated: "*I find it difficult to generalize so much, right? I imagine that it is a generic framework that will be useful in most cases*". In addition, participants noted that predicting and accounting for all possible failures in a given SoS context can be almost impossible, as unforeseen failures and behaviors can arise during the operational phase.

This diversity in perspectives underscores the challenge of developing a one-size-fits-all framework for the inherently varied nature of SoS. While some participants see ReViTA as highly adaptable, others highlight the nuanced and context-specific nature of SoS, suggesting that a universal solution might face limitations.

6.1.8. How is the Acceptance of ReViTA Among Professionals? (RQ1)

The overall reception of ReViTA among participants suggests a positive panorama, particularly in terms of its perceived usefulness and ease of use. The clarity and understandability of the activities emerged as notable strengths, alongside the acknowledged benefits of potential effort reduction.

A key insight from participant feedback is that ReViTA provides a systematic approach to problem-solving, integrating participants' prior insights and experiences. This is seen as a valuable contribution, fostering clear communication and assertive discussions among professionals, especially regarding fault

tolerance. Moreover, ReViTA was recognized as a facilitator for planning and a means to reduce the likelihood of errors in SoS management.

An important point was highlighted: the need for the development of appropriate tools to implement ReViTA's activities, particularly for large-scale or critical SoS. This was pointed out as a challenge to the framework's practical implementation.

6.2. Study #2

Study #2 involved four professionals who applied the ReViTA framework to a real-world case. They used a tool that supports the application of the framework. After the participants received training about the ReViTA framework and the tool, we asked them to:

1. Perform *Mission Decomposition* using the mKAOS modeler included in the tool;
2. Perform *Specification of Mission Requirements* for each individual mission defined in Mission Decomposition;
3. Perform the *Specification of States of Interest* of the constituent systems;
4. Perform the *Specification of Observation Points* of constituent systems;
5. Perform the *Specification of Mission Requirements for TAC*, which can be relaxed in relation to the desirable architectural configuration;
6. Use mKAOS modeler to perform *TAC Description*.

For feasibility reasons, the tool offers a simplified framework instantiation, as mentioned in Section 4. Except for the "*Mission Decomposition*" and "*TAC Description*" activities, which an mKAOS modeler supported, all other activities were conducted with textual descriptions. All the information provided by the participants was organized in a dashboard. This dashboard serves as a guide for professionals to design fault-tolerance-oriented reconfigurations. Figure 8 provides a summary of the participants' responses regarding the statements listed in Table 5.

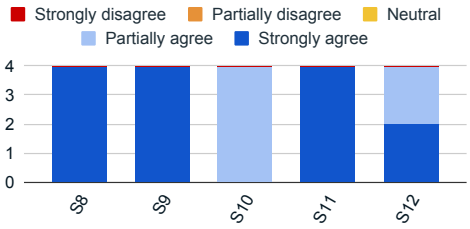

Figure 8: *Study #2* - Participants' responses.

6.2.1. Learning Experience (S8)

After utilizing the framework, all participants reported a positive learning experience. P15 and P17 specifically highlighted a modest learning curve, suggesting that the framework's comprehension was accessible. According to P17:

"*I learned a lot in a short time, and I think this is due to the fact that the activities are very clear regarding the expected result, in addition to being interconnected*". P15 reinforced the value of the framework's structured design, presented in a step-by-step format. This structure emerged as a significant facilitator in the learning process, contributing to participants' understanding and application of the framework.

Additionally, both P15 and P16 underscored the importance of prior familiarity with the scenario or context in which the framework is employed. This existing familiarity played a crucial role in easing the learning process, fostering a more intuitive grasp of the framework's intricacies and practical application.

Despite the positive learning experiences, participants identified a challenge regarding the Mission Decomposition step. They argued that breaking down missions to a level of granularity that a single constituent system can fulfill is a complex activity. Participants found this aspect non-trivial, indicating that it demanded substantial effort and understanding. Even with a comprehensive understanding of the SoS, achieving the right level of mission granularity remained a formidable challenge for all participants. These insights reinforce the need for specific tools to support the framework activities.

6.2.2. Clarity and Understanding (S9)

All participants strongly agreed that the clarity and understandability of ReViTA's purposes and activities are positive aspects. However, they faced some challenges in familiarizing themselves with the specific terminology used within the framework. This is explicit in P16 argument: "*It took me some time to adapt to these terms. They cause a bit of confusion*". P17 reinforced this perception: "*Regarding the terms, I had a lot of doubts. I am not familiar with this terminology. If you ask me now, I will have to stop and think about what you are talking about*".

Nonetheless, they expressed optimism that through continued use and experience, familiarity would be gained, enabling confident discussions about the framework with others. This feedback offers practical considerations for refining the framework's terminology to achieve a balance between specificity and user-friendly accessibility.

6.2.3. Ease of Use (S10)

The participants partially agreed that ReViTA is easy to use. They pointed out challenges that emerged in specific activities, highlighting both strengths and areas for improvement in the framework. For example, P16 underscored the critical nature of the mission decomposition process, investing significant effort to ensure its correctness. This participant acknowledged that a meticulous execution of this step was pivotal, as any compromise in mission decomposition could impact the subsequent processes. Once past this step, P16 found the remainder of the activities easier. On the other hand, P15 spent considerable time on "*TAC Description*" activity. Despite its time-consuming nature, P15 perceived the activity as relatively simple.

P17 faced challenges in performing activities, attributing difficulties more to the specific scenario than the framework itself. Challenges included achieving an adequate level of detail in the SoS, particularly in specifying the observation points and states of interest. The participant expressed concerns regarding potential oversights and encountered delays due to a lack of complete knowledge of some constituent systems. This knowledge gap necessitated consultation with another department to gain a better understanding, which subsequently caused delays in the process. Despite these challenges, P17 did not perceive them as negative aspects of the framework but rather as reflections of the complexity inherent in applying ReViTA to very large SoS. Emphasizing that the framework is easy to understand, P17 highlighted the substantial effort required in real-life applications. Moreover, P17 suggested that addressing these challenges would necessitate more specific tools to support tasks in large-scale SoS, indicating a need for additional resources for effective implementation in complex scenarios.

6.2.4. Usefulness (S11)

All participants strongly agreed on the framework's utility in their daily tasks. They acknowledged its role in supporting the design of fault-tolerant reconfigurations but also highlighted additional benefits it brings to their operations. P15, in particular, underscored the framework's ability to facilitate communication. One of the ways the framework enhances communication, as pointed out by P15, is within the SoS operations team. This team often needs to stay vigilant and responsive to changing objectives - a situation uncommon in their work. The framework, with its structured approach and clear guidelines, aids in keeping the team aligned and adaptable to these changes.

Furthermore, P15 emphasized the value of the documentation provided by the framework. This documentation is crucial in engaging with the sectors responsible for the constituent systems. These sectors often face challenges with effective communication and alignment with the SoS objectives. The framework's documentation can help bridge this communication gap and ensure better alignment with the SoS objectives.

P15 and P17 considered that the framework tends to produce more significant benefits when applied continuously, suggesting that its value increases with consistent use over time. They highlighted that, when applied continuously, ReViTA allows for ongoing information addition and constant evaluation of operations. Furthermore, this approach would allow more effective monitoring and adjustment of the SoS, enhancing ReViTA overall usefulness.

6.2.5. Impact on Job Performance (S12)

Two participants strongly agreed that utilizing the framework would enhance their job efficiency. P17 and P18, who expressed partial agreement, emphasized that the initial application of the framework requires significant effort. However, they also highlighted that the effort tends to decrease over time as the framework is applied and refined, leading to an overall increase in work efficiency.

P16 focused on the positive impact of the framework on daily operations. Given the need for the SoS team to maintain reliability in the face of poten-

tial disruptions in the constituent systems, the framework's emphasis on fault tolerance was a key benefit.

P18 pointed out an additional benefit of the framework: its capacity to identify areas for infrastructure improvement. According to P18, the framework offers a comprehensive scenario view. It provides a complete understanding of the functioning of the constituent systems that aids in avoiding failures in the SoS. Beyond focusing on fault tolerance, the framework also provides valuable information that can enhance the overall infrastructure. Analyzing the proposed scenario makes it possible to identify areas that need improvement, irrespective of the implementation of fault tolerance. This approach assists in pinpointing weaknesses, particularly the most vulnerable links, which can then be improved to ensure efficiency and robustness in the SoS. Thus, while the framework's primary goal is to work on fault-tolerant reconfiguration, it rather contributes to the broader objective of infrastructure enhancement.

6.2.6. Is ReViTA Feasible From a Practical Perspective? (RQ2)

Drawing from the findings of Study 2, we gathered insightful perspectives on the practical feasibility of ReViTA. One prominent consideration that emerged centers around the essential need for dedicated tools to execute the framework activities successfully. This need becomes particularly important in the context of large-scale SoS, which involve intricate interplay between multiple missions and constituent systems.

The participants' acceptance of ReViTA and recognition of its potential to enhance their work can be seen as a positive aspect toward its practical applicability. The acknowledgment of ReViTA's value extends beyond its immediate impact, emphasizing the potential for progressive refinement and improvement of results through consistent application. This highlights the dynamic nature of ReViTA, suggesting that with continuous use, it cannot only address immediate SoS reliability issues but also foster advancements in the management of fault tolerance in SoS.

7. Discussion

In this section, we discuss the main findings and the implications of our research and propose how ReViTA can be used by researchers and practitioners.

7.1. Main Findings and Lessons Learned

Along the process to design and evaluate ReViTA, we gathered insights and reflections about the framework as in the following:

SoS fault tolerance requires the involvement of domain experts. Both evaluation studies highlighted the importance of a deep understanding of the SoS domain while dealing with fault tolerance. The participants stated that this is fundamental for adequately applying ReViTA and ensuring accuracy in performing its activities since each SoS operates under a distinct set of conditions that may be technical, environmental, and operational. Hence, solid

and comprehensive domain knowledge improves efficiency and accuracy when implementing ReViTA.

The value of ReViTA increases with continuous use. The continuous application of ReViTA is necessary not only to allow fault tolerance mechanisms to accommodate the evolution of the SoS missions over time. From the participants' comments, it becomes apparent that the framework's value increases with continual use. As the professionals gain more experience with the framework, they can more effectively manage the complexities of SoS, leading to improved system performance and mission fulfillment. The continuous application of the framework enables the constant addition of information and ongoing evaluation of operations, which could result in improved effectiveness of fault tolerance oriented reconfigurations.

The use of ReViTA produces valuable documentation. ReViTA provides a comprehensive outlook on SoS fault tolerance, highlighting the responsibilities of constituent systems, potential disturbances, expected behaviors, and related countermeasures. Participants stated that teams working on SoS often must deal with complex scenarios involving multiple interconnected constituent systems, which could lead to misunderstandings or miscommunications. Using ReViTA produces valuable outcomes, helping to maintain team members on the same page and enabling a more seamless integration of efforts.

The use of ReViTA increases with the scale of SoS. Applying ReViTA to large-scale SoS introduces a high level of complexity due to the multitude of interconnected elements such as mission requirements, states of interest, and TAC, each with their specific characteristics and interactions. The situation becomes more complicated when dealing with a large-scale SoS, as the quantity of these elements and interactions can multiply exponentially. Such circumstances create a highly complex scenario requiring careful attention.

ReViTA requires the involvement of decision-makers. During *Study #2*, some participants reported that the decision to employ specific constituent systems in TAC would require approval from higher management, as it would entail extra costs for the institution. Hence, we recognize that the active involvement of decision-makers is critical, as they have the authority to establish strategies and define priorities. As observed by the participants, describing TAC can encompass financial considerations, such as constituent systems' acquisition and maintenance costs. However, other factors, such as privacy and security, may influence TAC Description. In this context, it falls to the decision-makers to assess associated risks and direct the allocation of resources to improve SoS fault tolerance. Consequently, they play an essential role in reinforcing reliability.

Regarding **lessons learned**, during the conduction of the evaluation studies, we recognized the importance of including domain experts in the evaluation process. While professionals with a background in Information Technology (IT) bring valuable technical expertise, they may lack a comprehensive understanding of domain-specific particularities, which could be important in evaluating the adaptability of the framework. Moreover, the perspectives that IT professionals consider may overlook crucial domain-specific aspects. Domain experts, with

their specialized knowledge, could provide additional insights. They can ensure the framework does not overlook non-IT technical aspects crucial for adequate fault tolerance.

Another lesson learned is regarding the need for the development of a tool that participants would be able to use. Creating a too-specific tool could limit our participant selection, requiring specific knowledge of particular approaches. On the other hand, the tool, although simplified, should be capable of encompassing all the framework's activities clearly so that participants can express their opinions on ReViTA without being limited by the tool.

Finally, it is important to highlight the difference between our work and the approach proposed by Maia *et al.* [36], in which the authors proposed a three-step approach that uses feedback loops to identify and handle "defiant" constituent systems, which are those that cannot achieve SoS requirements at a given time. The authors presented a process to identify such constituents. Then, wrappers are input into the challenging constituent systems to support changes in the behavior of the challenging components while maintaining satisfaction with the overall system requirements. Finally, feedback loops are used to monitor and ensure the self-adaptation of the SoS. Contrary to this proposal, which relies on the inclusion of wrappers on constituent systems, ReViTA does not modify constituent systems; instead, it proposes changes to the SoS architectural configuration, which may include new constituent systems or temporarily remove those that are producing disturbances in the SoS.

7.2. Implications for Practitioners and Researchers

This study presents implications for **practictioners**, as follows:

Improvement of SoS reliability. Using ReViTA can lead to an improvement in SoS reliability. By following the activities prescribed by the framework, professionals can identify and implement fault tolerance oriented reconfigurations as effective countermeasures, reducing the probability of disturbances and interruptions in SoS operations.

Improvement of stakeholders communication. ReViTA was perceived as a facilitator of effective communication among varied stakeholders. It can help promote a shared understanding regarding fault tolerance, clear communication of objectives, collaboration and coordination, improved decision-making, and stakeholder engagement.

Improvement of resource utilization. By offering a comprehensive view of SoS fault tolerance elements, professionals can make well-informed decisions, prioritize actions more efficiently, and coordinate their efforts more effectively. The outcomes of ReViTA activities can help professionals optimize resources (e.g., acquisition and maintenance of constituent systems) and minimize operational costs related to fault tolerance. By identifying high-risk areas and implementing appropriate reconfiguration strategies, professionals can avoid unnecessary expenses and maximize the operational efficiency of the SoS. Moreover, by using ReViTA, professionals can plan TAC, allowing professionals to anticipate and pre-authorize any additional costs incurred during critical and

time-sensitive situations. By clearly understanding the potential extra costs associated with reconfiguration, professionals can make informed decisions and allocate resources accordingly, ensuring faster recovery when it is essential.

During the conduction of this study, we also identified implications for **researchers**, as in the following:

Mission-oriented perspective for future research. The use of ReViTA provides a comprehensive view of the SoS, focusing on its mission. This comprehensive perspective can also facilitate in-depth analysis of other attributes such as security, performance, and scalability, among others. Additionally, it illustrates how these attributes contribute to achieving the SoS overall mission.

Adaptability to diverse SoS contexts. ReViTA can serve as a foundation for future research on adaptability in different SoS contexts. Researchers can explore how the framework can be tailored and customized to specific domains. This research can lead to developing domain-specific extensions or variations of the framework, enabling its effective application in a more significant range of SoS environments.

8. Threats and Limitations

8.1. Threats to Credibility and Reliability

Unlike quantitative studies, qualitative ones are typically more susceptible to threats to credibility rather than threats to validity [54, 55]. The matters of validity and reliability in qualitative research rely on the meticulousness, thoroughness, and honesty employed by the researchers throughout the data collection and analysis processes [56]. Thus, we outline the potential threats to external and internal credibility in the following.

Internal credibility refers to the credibility of interpretations and conclusions within the underlying setting or group [57]. In this study, interpretive validity is a potential threat to internal credibility, which describes the risk of researchers imposing their interpretations rather than understanding the participants' perspectives. We mitigated this threat by asking clear questions to participants and encouraging them to reflect deeply on their answers so that we could obtain a realistic interpretation of the collected information. Moreover, the interviews were performed face-to-face since this approach makes the participants more spontaneous in their answers [58]. Moreover, the interviews were conducted in Portuguese, which is the native language of the participants.

Regarding the analysis, we applied coding to the interview transcriptions. Coding is a systematic approach to interpreting and analyzing interview data, ensuring that all responses are evaluated consistently and reducing the risk of bias or interpretive errors.

The participants' lack of experience regarding SoS Engineering and the fact that all of them worked with a single scenario of SoS are threats to **external credibility**, which refers to the degree that the findings of a study can be generalized across different contexts [57]. Moreover, we understand that the number of participants in the studies is not representative enough to generalize

the results. We selected professionals with different backgrounds and experience implementing or operating an SoS to mitigate this. This contributed to a more significant variety of information with different perspectives. We also carried out studies from distinct perspectives. The first study focused on participants' perceptions of the framework through demonstration and explanation. In contrast, the second one took a more practical approach, with participants using the framework and offering their opinions based on a practical experience.

We acknowledge that how we framed our interviews could potentially influence participant responses, thereby affecting the **reliability** of our study. To mitigate this effect, we engaged in extensive discussions about the interview guide among the authors of this article and refined it further after conducting pilot interviews. During the interviews, the researcher paid close attention to aspects such as voice intonation and body language to guide the interview, posing additional questions to ensure a comprehensive collection of participants' perceptions. In addition, the composition of objective questions (using a 5-point Likert scale) and open-ended questions allowed for more accurate verification of responses. For instance, if there were contradictions between the two responses, the researcher sought to investigate and clarify further.

We provided a confidentiality agreement to the participants to ensure they felt comfortable expressing their opinions without worrying about repercussions, given that they provided opinions on a scenario related to their place of employment. The interviews were automatically transcribed and individually reviewed to correct transcription errors.

8.2. Limitations

It is essential to acknowledge that our evaluation is limited to only one specific scenario. While our findings and conclusions provide valuable insights into that particular context, validating the results in other contexts is crucial. By conducting studies in multiple contexts, we can assess the robustness and applicability of our findings across various scenarios. This helps establish a broader understanding of the framework's effectiveness and potential limitations in different real-world situations.

Furthermore, exploring different contexts can uncover additional insights and nuances that our studies might not capture. It can shed light on new challenges and contextual factors that may influence the implementation and outcomes of the framework. Hence, such additional validation can provide a more comprehensive understanding of the framework's capabilities and limitations.

Another limitation is that ReViTA does not apply to virtual SoS. Given their inherent lack of defined goals [1], the execution of the framework's activities becomes impracticable as ReViTA's applicability depends on clearly defined objectives, thus leaving virtual SoS outside its scope of applicability.

9. Conclusion

Concerns on SoS fault tolerance are not limited to failures in constituent systems. It also encompasses undesirable behaviors resulting from their inde-

pendence. Moreover, as the design of SoS is inherently opportunistic, fault tolerance mechanisms for SoS should follow suit.

Current literature partially addresses the SoS dynamism and the use heterogeneous redundancies for fault tolerance in SoS. While some studies advocate for it, they overlook that the primary architectural configuration is often the best fit for its specific needs, considering factors such as costs, security, and performance, to mention a few. Moreover, using heterogeneous redundancies demands precise information that feeds the monitoring and reconfiguration processes, and there is a lack of solutions to support this effectively. ReViTA seeks to address this issue by providing structured activities that provides effective information to these processes aiming at fault tolerance, considering the dynamic and opportunistic nature of SoS design.

By using ReViTA, professionals can better understand SoS critical points, and the fault tolerance countermeasures. This enables them to design fault-tolerant reconfigurations in a more informed and effective manner. Furthermore, our framework facilitates more effective communication among the SoS team regarding fault tolerance, ensuring everyone is on the same page. It also supports decision-making regarding the utilization of resources.

As part of future work, it is imperative to highlight the ongoing challenges associated with the identification and selection of appropriate scenarios for experimentation in SoS. The inherent complexity and uniqueness of SoS often make it challenging to obtain a diverse and representative set of scenarios for comprehensive evaluation. Recognizing this challenge, we emphasize the need for further studies with an expanded participant base to enhance the robustness of our findings. Moreover, we can investigate how the framework might be adapted or extended to meet specific domain requirements.

Moreover, future work should consider additional investigations for collaborative SoS. Unlike directed and acknowledged SoS, these types of SoS lack central management that coordinates and influences constituent systems. This could limit the availability of observation points within constituent systems, potentially restricting the application of ReViTA in such contexts. Finally, developing tools to support ReViTA activities is crucial, particularly for large-scale SoS, due to the exponential nature of potential combinations of missions, requirements, states of interest, and TAC.

Acknowledgements

This study was funded by UNIRIO (DPq/PPQ 2022 and 2023), FAPERJ (211.583/2019), CAPES, CNPq (313245/2021-5), and FAPESP (2015/24144-7, 2017/06195-9, 2023/00488-5). The first author also thanks the Federal University of Juiz de Fora.

References

[1] M. W. Maier, Architecting principles for systems-of-systems, Systems Engineering 1 (4) (1998) 267–284.

[2] R. Raman, A. Murugesan, Framework for complex sos emergent behavior evolution using deep reinforcement learning, INCOSE International Symposium 32 (1) (2022) 809–823.

[3] F. H. Ferreira, E. Y. Nakagawa, R. P. Santos, Reliability in software-intensive systems: Challenges, solutions, and future perspectives, in: Euromicro SEAA, Palermo, Italy, 2021, pp. 54–61.

[4] W. Blischke, D. Murthy, Reliability: Modeling, Prediction, and Optimization, 1st Edition, Wiley, 2000.

[5] B. P. Zeigler, M. Redding, P. J. Boyers, E. L. Carter, Model-based systems-of-systems healthcare: Coordinating the coordinators, in: A. M. Madni, B. Boehm, D. Erwin, M. Moghaddam, M. Sievers, M. Wheaton (Eds.), Recent Trends and Advances in Model Based Systems Engineering, Springer Int. Publishing, 2022, pp. 515–527.

[6] I. Phillips, R. Kenley, Verification of intelligent transportation systems: Challenges and possibilities, in: System of Systems Engineering Conference (SOSE), Rochester, USA, 2022, pp. 127–131.

[7] K. Sahu, R. Srivastava, Revisiting software reliability, in: International Conference on Data Management, Analytics Innovation, Vol. 808, Springer Singapore, 2019, pp. 221–235.

[8] F. H. C. Ferreira, E. Y. Nakagawa, R. P. Santos, Towards an understanding of reliability of software-intensive systems-of-systems, Information and Software Technology 158 (2023) 107186.

[9] P. G. Teixeira, B. G. A. Lebtag, R. P. dos Santos, M. Kassab, F. Horita, V. V. G. Neto, Externalizing requirements for achieving operational independence in systems-of-systems: A mapping study, in: 2023 18th Annual System of Systems Engineering Conference (SoSe), 2023, pp. 1–6.

[10] G. Rebovich, John Wiley & Sons, 2009, Ch. Enterprise System of Systems.

[11] M. Imamura, F. H. Ferreira, J. C. Fernandes, R. Santos, System-of-systems reliability: An exploratory study in a Brazilian public organization, in: Brazilian Symposium on Information Systems (SBSI), Uberlândia, Brazil, 2021, pp. 1–8.

[12] J. Boardman, B. J. Sauser, System of systems - the meaning of of, in: SoSE, Los Angeles, USA, 2006, pp. 1–6.

[13] J. Dahmann, G. Rebovich, J. Lane, Systems engineering for capabilities, CrossTalk 21 (2008) 7.

[14] P. Uday, K. Marais, Exploiting Stand-in Redundancy to Improve Resilience in a System-of-Systems (SoS), in: Conference on Systems Engineering Research (CSER), Atlanta, USA, 2013, pp. 532–541.

[15] F. Petitdemange, I. Borne, J. Buisson, Design process for system of systems reconfigurations, Systems Engineering 24 (2) (2021) 69–82.

[16] F. Petitdemange, I. Borne, J. Buisson, Approach based patterns for system-of-systems reconfiguration, in: Int. Workshop on Software Engineering for Systems-of-Systems (SESoS), Florence, Italy, 2015.

[17] M. Lyu, Software reliability engineering: A roadmap, in: Future of Soft. Eng. (FOSE), Minneapolis, USA, 2007, pp. 153–170.

[18] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Transactions on Dependable and Secure Computing 1 (1) (2004) 11–33.

[19] A. Aviziens, Fault-tolerant systems, IEEE Transactions on Computers C-25 (12) (1976) 1304–1312.

[20] M. R. Lyu (Ed.), Handbook of Software Reliability Engineering, McGraw-Hill, Inc., USA, 1996.

[21] J. Cook, Multi-state reliability requirements for complex systems, in: RAMS, Las Vegas, USA, 2008, pp. 317–321.

[22] A. Romanov, M. Romanov, A. Kharchenko, Fpga-based control system reconfiguration using open source software, in: Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg and Moscow, Russia, 2017, pp. 976–981.

[23] S. Forte, T. Dickopf, S. Weber, J. C. Göbel, Towards sustainable systems reconfiguration by an iot-driven system of systems engineering lifecycle approach, Procedia CIRP 105 (2022) 654–659.

[24] T. Wong, M. Wagner, C. Treude, Self-adaptive systems: A systematic literature review across categories and domains, Information and Software Technology 148 (2022) 106934.

[25] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Di Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, J. Whittle, Software Engineering for Self-Adaptive Systems: A Research Roadmap, Springer Berlin Heidelberg, 2009, pp. 1–26.

[26] P. Arcaini, E. Riccobene, P. Scandurra, Modeling and analyzing mape-k feedback loops for self-adaptation, in: International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Florence, Italy, 2015, pp. 13–23.

[27] Z. Andrews, J. Fitzgerald, R. Payne, A. Romanovsky, Fault Modelling for Systems of Systems, in: International Symposium on Autonomous Decentralized Systems (ISADS), Mexico City, Mexico, 2013, pp. 1–8.

[28] Z. Andrews, R. Payne, A. Romanovsky, A. Didier, A. Mota, Model-based development of fault tolerant systems of systems, in: Int. Systems Conference (SysCon), Orlando, USA, 2013, pp. 356–363.

[29] Z. Andrews, C. Ingram, R. Payne, A. Romanovsky, J. Holt, S. Perry, Traceable Engineering of Fault-Tolerant SoSs, INCOSE International Symposium 24 (1) (2014) 258–273.

[30] C. Ingram, Z. Andrews, R. Payne, N. Plat, Sysml fault modelling in a traffic management system of systems, in: Int. Conf. on System of Systems Engineering (SOSE), Glenelg, Australia, 2014, pp. 124–129.

[31] B. Mokhtarpour, J. T. Stracener, Mission reliability analysis of phased-mission systems-of-systems with data sharing capability, in: RAMS, Palm Harbor, USA, 2015, pp. 1–6.

[32] B. Wudka, C. Thomas, L. Siefke, V. Sommer, A reconfiguration approach for open adaptive systems-of-systems, in: International Symposium on Software Reliability Engineering Workshops (ISSREW), Coimbra, Portugal, 2020, pp. 219–222.

[33] N. Bhardwaj, P. Liggesmeyer, A conceptual framework for safe reconfiguration in open system of systems, in: Int. Workshop on Software Engineering for Systems-of-Systems, 2018, pp. 17–20.

[34] D. Weyns, J. Andersson, On the challenges of self-adaptation in systems of systems, in: International Workshop on Software Engineering for Systems-of-Systems (SESoS), Montpellier, France, 2013, p. 47–51.

[35] S. Wätzoldt, H. Giese, Modeling collaborations in adaptive systems of systems, in: European Conference on Software Architecture Workshops (ECSAW), Dubrovnik, Croatia, 2015.

[36] P. H. Maia, L. Vieira, M. Chagas, Y. Yu, A. Zisman, B. Nuseibeh, Cautious adaptation of defiant components, in: International Conference on Automated Software Engineering (ASE), San Diego, USA, 2019, pp. 974–985.

[37] F. Lonetti, V. de Oliveira Neves, A. Bertolino, Designing and testing systems of systems: From variability models to test cases passing through desirability assessment, Journal of Software: Evolution and Process 34 (10) (2022) e2427.

[38] R. J. Wieringa, What Is Design Science?, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 3–11.

[39] R. Khan, Secure software development: a prescriptive framework, Computer Fraud Security 2011 (8) (2011) 12–20.

[40] F. H. Ferreira, E. Y. Nakagawa, R. P. Santos, Towards an understanding of reliability of software-intensive systems-of-systems, Information and Software Technology 158 (2023) 107186.

[41] M. C. Harrell, M. A. Bradley, Data Collection Methods: Semi-Structured Interviews and Focus Groups, RAND Corp., Santa Monica, CA, 2009.

[42] W. C. Adams, Conducting Semi-Structured Interviews, John Wiley Sons, Ltd, 2015, Ch. 19, pp. 492–505.

[43] F. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Quartely 13 (3) (1989) 319–340.

[44] J. Corbin, A. Strauss, Basics of qualitative research: Techniques and procedures for developing grounded theory, 3rd Edition, SAGE, 2008.

[45] I. Cherfa, N. Belloir, S. Sadou, R. Fleurquin, D. Bennouar, Systems of systems: From mission definition to architecture description, Systems Engineering 22 (6) (2019) 437–454.

[46] B. Zelalem Mihret, E. Jee, Y.-M. Baek, D.-H. Bae, A collaboration policy model for system of systems, in: International Conference on System of Systems Engineering (SoSE), Paris, France, 2018, pp. 1–8.

[47] E. Silva, T. Batista, F. Oquendo, A mission-oriented approach for designing system-of-systems, in: System of Systems Engineering Conference (SoSE), San Antonio, USA, 2015, pp. 346–351.

[48] J. Hu, L. Huang, X. Chang, B. Cao, A model driven service engineering approach to system of systems, in: International Systems Conference (SysCon), Ottawa, Canada, 2014, pp. 136–145.

[49] E. Silva, E. Cavalcante, T. Batista, F. Oquendo, F. C. Delicato, P. F. Pires, On the characterization of missions of systems-of-systems, in: European Conference on Software Architecture Workshops (ECSAW), 2014, pp. 1–8.

[50] H. Lampesberger, M. Rady, Correct Software in Web Applications and Web Services, SpringerLink, 2015, Ch. Monitoring of Client-Cloud Interaction, pp. 177–228.

[51] J. Postel, RFC 792 - Internet Control Message Protocol, Request for Comments - Internet Engineering Task Force (1981).

[52] J. El Hachem, Z. Y. Pang, V. Chiprianov, A. Babar, P. Aniorte, Model driven software security architecture of systems-of-systems, in: 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), Hamilton, New Zealand, 2016, pp. 89–96.

[53] B. Mokhtarpour, J. Stracener, A Conceptual Methodology for Selecting the Preferred System of Systems, IEEE Systems Journal 11 (4) (2017) 1928–1934.

[54] M. Greiler, M. Storey, A. Noda, An actionable framework for understanding and improving developer experience, IEEE Transactions on Software Engineering 49 (04) (2023).

[55] B. B. Ribeiro, C. Costa, R. Pereira dos Santos, Understanding and analyzing factors that affect merge conflicts from the perspective of software developers, Journal of Software Engineering Research and Development 10 (2022) 12:1 – 12:17.

[56] C. Robson, Real World Research - A Resource for Social Scientists and Practitioner-Researchers, Oxford: Blackwell Publishing, 2002.

[57] A. J. Onwuegbuzie, N. L. Leech, Validity and Qualitative Research: An Oxymoron?, Quality Quantity 41 (2) (2007) 233–249.

[58] R. Opdenakker, Advantages and disadvantages of four interview techniques in qualitative research, Forum Qualitative Sozialforschung / Forum: Qualitative Social Research 7 (4) (2006).