UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

A compositional approach for Systems-of-Systems safety analysis

Samuel de Souza Lopes

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Samuel de Souza Lopes

A compositional approach for Systems-of-Systems safety analysis

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Rosana Teresinha Vaccare Braga

USP – São Carlos
January 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Lopes, Samuel de Souza
L864c A compositional approach for Systems-of-Systems safety analysis / Samuel de Souza Lopes; orientadora Rosana Teresinha Vaccare Braga. -- São Carlos, 2023.
123 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. Software Engineering. 2. System-of-Systems. 3. Model-Based Systems Engineering. 4. Safety Analysis. I. Braga, Rosana Teresinha Vaccare, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Samuel de Souza Lopes

Uma abordagem composicional para análise de segurança de Sistemas-de-Sistemas

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Prof. Dra. Rosana Teresinha Vaccare Braga

USP – São Carlos
Janeiro de 2023

For God, my parents, my girlfriend, my family, and my friends.

ACKNOWLEDGEMENTS

Throughout my life, several people make the paths I follow less torturous. Some were with me during some stages of my life, while others remain forever. In this context, God has always been with me and all I have is thanks to Him. So, first of all, I want to thank God for being directly responsible for all my joys, as well as for giving me enough energy and wisdom to overcome all the obstacles that have appeared in my life.

Concerning the people who are with me always, I thank my parents for all the care and dedication provided throughout my life. I also thank the dear members of my family and my special friends, with an emphasis on my girlfriend and my sisters, for their unconditional care and support.

Regarding my academic development, I would like to thank my master's advisor, Prof. Dr. Rosana Teresinha Vaccare Braga, as well as my supervisor Prof. Dr. Andre Luiz de Oliveira for contributing directly to my evolution as a professional, to express constructive criticism and praise for the research developed during my master's degree. In addition, I would like to thank my graduation advisor, Prof. Dr. Rogéria Cristiane Gratão de Souza, for having contributed to my academic development. All the support provided by both was crucial and I will always be grateful for that.

I want to thank my friends from *Software Engineering Lab* (LaBES), especially those who helped me during my master's degree. I also thank the *University of São Paulo* (USP) and its employees for providing me with the opportunity to study at a university of great national and international prestige. I thank the academy jury members for making constructive criticisms about this research. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

“Tudo é possível para quem tem fé.”
(Marcos 9:23)

ABSTRACT

LOPES, S. S. **A compositional approach for Systems-of-Systems safety analysis**. 2023. 123 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

The Systems-of-Systems life cycle is challenging due to inherent System-of-Systems characteristics, such as autonomy, belonging, connectivity, diversity, and emergency. Different hazardous behaviors may arise from these characteristics, preventing the System-of-Systems from performing its mission. A *hazard* is a potential condition that can cause injury, illness, or death to personnel, damage to or loss of a system, equipment, or property, or damage to the environment. At the System-of-Systems-level, hazards can emerge from interactions between Constituent Systems and inside a given system. System-of-Systems hazardous behaviors can propagate throughout the Constituent Systems, and managing them is complex, time-consuming, and error-prone. Performing System-of-Systems safety analysis is still challenging since existing safety analysis techniques and tools do not consider the its inherent characteristics that can emerge throughout the System-of-Systems life cycle. In this context, this work intends to support safety analysis at the System-of-Systems-level to define which Constituent Systems meet the systems' safety properties to be incorporated into the System-of-Systems operation. This objective has been reached by proposing an approach that intends to adapt existing compositional techniques to enable semi-automated support for System-of-Systems safety analysis, as well as a meta-model to support System-of-Systems design and safety analysis, which consists of a structured way to model the information regarding System-of-Systems and its Constituent Systems to perform systems safety analysis. The approach was evaluated through an illustrative study of a System-of-Systems from the automotive domain, which provided evidence that System-of-Systems safety analysis can be performed at the System-of-Systems-level.

Keywords: Software Engineering, System-of-Systems, Model-Based Systems Engineering, Safety Analysis.

RESUMO

LOPES, S. S. **Uma abordagem composicional para análise de segurança de Sistemas-de-Sistemas**. 2023. 123 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

O ciclo de vida dos Sistemas-de-Sistemas é desafiador devido às características inerentes do Sistema-de-Sistemas, como autonomia, pertencimento, conectividade, diversidade e emergência. Diferentes comportamentos perigosos podem surgir dessas características, impedindo que o Sistema-de-Sistemas cumpra sua missão. Um *perigo* é uma condição potencial que pode causar ferimentos, doenças ou morte de pessoas, danos ou perda de um sistema, equipamento ou propriedade, ou danos ao meio ambiente. No nível do Sistema-de-Sistemas, os perigos podem surgir das interações entre os Sistemas Constituintes e dentro de um determinado sistema. Comportamentos perigosos do sistema de sistemas podem se propagar por todos os Sistemas Constituintes, e gerenciá-los é complexo, demorado e propenso a erros. Realizar a análise de segurança de Sistemas-de-Sistemas ainda é um desafio, pois as técnicas e ferramentas de análise de segurança existentes não consideram as características inerentes do Sistema-de-Sistemas que podem surgir ao longo do ciclo de vida do Sistemas-de-Sistemas. Neste contexto, o objetivo do presente trabalho é apoiar analise de segurança no nivel de Sistemas-de-Sistemas para definir quais Sistemas Constituintes atendem os critérios de propriedade de segurança de sistemas para serem incorporados a operação de Sistemas-de-Sistemas Este objetivo foi alcançado através da proposta de uma abordagem com a intenção de adaptar as técnicas de composição existentes para permitir suporte semi-automatizado para análise de segurança de Sistemas-de-Sistemas, bem como um meta-modelo para apoiar o projeto e análise de segurança de Sistemas de Sistemas, que consiste em uma forma estruturada de modelar as informações referentes ao Sistema-de-Sistemas e seus Sistemas Constituintes para realizar a análise de segurança dos sistemas. A abordagem foi avaliada através de um estudo ilustrativo de um Sistema-de-Sistemas do domínio automotivo, que forneceu evidências de que a análise de segurança do Sistema-de-Sistemas pode ser realizada no nível do Sistema-de-Sistemas.

Palavras-chave: Engenharia de Software, Sistema-de-Sistemas, Engenharia de Sistemas Baseado em Modelos, Análise de Perigos.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

SoSSafe	SoS Safety Analysis
AADL	Architecture Analysis and Design Language
ADAS	Advanced Driver-Assistance System
AFS	Anti-Flood System
AMADEOS	Architecture for Multi-criticality Agile Dependable Evolutionary Open System-of-Systems
ASIL	Automotive Safety Integrity Level
BBN	Bayesian Belief Network
CA	Criticality Analysis
CAS	Cooperative Autonomous System
CCF	Common Cause Failure
CoS	Coalition of Systems
CPS	Cyber-Physical System
CS	Constituent System
DAL	Development Assurance Level
DEIS	Dependability Engineering Innovation for cyber-physical Systems
E/E	Electrical and Electronic
EC	Exlusion Criteria
EMF	Eclipse Modeling Framework
EMV2	Error Model Annex, Version 2
ESP	Electronic Stability Program
EUROCAE	European Organisation for Civil Aviation Equipment
EV	Ego Vehicle
FHA	Functional Hazard Assessment
FLA	Failure Logic Analysis
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Modes, Effects, and Criticality Analysis
FPTN	Failure Propagation and Transformation Calculus
FPTN	Failure Propagation and Transformation Notation
FTA	Fault Tree Analysis
HARA	Hazard Analysis and Risk Assessment

HAZOP	HAZard and OPerability Studies
Hip-HOPS	Hierarchically Performed Hazard Origin and Propagation Studies
HISoS	Hazards In Systems-of-Systems
IC	Inclusion Criteria
ICSA-C	IEEE International Conference on Software Architecture Companion
IDS	Integrated Deep Water System
ISO	International Organization for Standardization
kg	kilogram
MOF	Meta-Object Facility
NEC	Network Enabled Capability
ODE	Open Dependability Exchange
PSSA	Preliminary System Safety Analysis
QC	Quality Criteria
QM	Quality Management
RPN	Risk Priority Number
RQs	Research Questions
SMRQ	Systematic Mapping Research Question
SOA-ML	Service-Oriented Architecture Modeling Language
SoS	System-of-Systems
SoSE	System-of-Systems Engineering
STAMP	Systems-Theoretic Accident Model and Processes
SysML	Systems Modeling Language
TLA	Traffic Light Assistant
UML	Unified Modelling Language
V2V	Vehicle-to-Vehicle

CONTENTS

CHAPTER

1

INTRODUCTION

1.1 Context

System-of-Systems (SoS) is a set of operational and managerial independent systems in which each system is called Constituent System (CS). CSs work together to achieve a behavior at the SoS-level that any CS cannot provide in isolation (MAIER, 1998). Due to operational and managerial characteristics, CSs have their own development, management goals, and resources and may be situated in different geographical locations around the globe (LOPEZ, 2006; WOJCIK; HOFFMAN, 2006). SoSs have been used in different application domains, such as intelligent transportation (MATHEW, 2020), aerospace (GUARINIELLO *et al.*, 2019), energy (GUNAWAN *et al.*, 2017), healthcare (LEITE; SCHNEIDER; ADLER, 2018), and military (CHEN; UNEWISSE, 2017). A smart city is a typical application domain where SoS concepts have been applied. In smart cities, citizens consume energy, materials, and services to catalyze economic development and improve their quality of life through government services, transportation, healthcare, energy, and water systems that interact to achieve effectiveness and efficiency for citizens (ELSHENAWY; ABDULHAI; EL-DARIEBY, 2018; CAVALCANTE *et al.*, 2017).

There has been an increasing role of safety concepts in the context of SoS applications. Smart cities can be considered safety-critical systems since government or healthcare systems failures may lead to catastrophic damages at the SoS-level, such as loss of human life or environmental damage. One failure example of these CSs could be due to a lack of oxygen in hospitals in the city, causing human injuries or even human deaths. Such failure can be caused by a technical failure from the healthcare system (SHABAN *et al.*, 2022), or embezzlement of health funds made illegally from the government system (MALTA; STRATHDEE; GARCIA, 2021). Therefore, approaches to support SoS safety analysis are necessary because of the critical nature of SoS, in which SoS hazards caused by a combination of failures to at least one of its CSs may lead to catastrophic damages to the property, environment, injuries, or loss of human

life (ALEXANDER; HALL-MAY; KELLY, 2004).

Safety analysis seeks to identify and address safety requirements to minimize or eliminate safety risks during all phases of the development life cycle (DEZFULI *et al.*, 2011). Besides smart cities cited before, several application domains have SoS operating in safety-critical contexts can be mentioned. For example, there is the vehicle platooning system in the automotive domain, in which a lead vehicle is followed by other vehicles that are driven semi-autonomously at a very short distance between each other and possibly coordinated by a higher level controller (KOBETSKI; AXELSSON, 2017). In the defense domain, there is the anti-missile system composed of satellite, radar, warning center, brigade post, battalion post, and cannon company, which intends to implement multiple functions such as warning detection, intelligence reconnaissance, command/control, fire fighting, and communication (MENGMENG *et al.*, 2018).

Safety analysis approaches can be used to support safety life-cycle activities. Such techniques are divided into two categories: traditional and model-based. Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA) are well-established traditional techniques that can be used to identify potential faults in a system. FTA is a deductive technique because it considers an analysis from a top event (typically a system failure) and then deducts the component failure events which caused the top event. In contrast, FMEA is an inductive technique since the analysis starts from component failure events and identifies their potential effects on the system. The output of both techniques helps correct or prevent the analyzed failures. FTA output is fault trees, whereas FMEA tables are FMEA output. Fault trees are suitable to identify how SoS hazard effects propagate throughout CSs. FMEA tables are derived from the fault trees. They help to identify how CSs can fail directly, i.e., when a failure in a CS will lead to the occurrence of an SoS hazard, or indirectly, i.e., when a CS failure, in conjunction with one or more failures in other CSs, will cause an SoS hazard (PAPADOPOULOS *et al.*, 2011). An issue with traditional safety analysis approaches is that most existent ones are manual to be performed even with the support of existing tools to guide the analysis. To support the automation of traditional safety analysis techniques and enable automatic synthesis of fault trees and FMEA tables, compositional safety analysis techniques and tools have been proposed to integrate safety analysis with the system design process (DELANGE; FEILER, 2014; PAPADOPOULOS *et al.*, 2011).

Compositional safety analysis techniques are a category of model-based safety analysis approaches which provide formal and semi-formal languages to support specification, composition, and analysis of component failure behavior based on safety information regarding its components (SHARVIA *et al.*, 2016). Failure Propagation and Transformation Notation (FPTN) is a classic compositional safety analysis technique and probably the first one which describes the failure behavior of a given system graphically by describing the generation and propagation of component failures in the system. These component modules are connected via inputs and outputs to other modules, allowing the combination and propagation of failures from one module to

another. They can be composed into subsystems used to build a system hierarchy. In turn, Failure Propagation and Transformation Calculus (FPTN) is an extension of FPTN, which intends to link the failure model to the architectural model so that all dependencies are identified and maintained. FPTC defines different failure classes (like *omission*, *commission*, and *value* errors), which are directly specified in annotations in the system model's components. The inputs and outputs of those components are then used to transmit failure information to the rest of the system by using a set of expressions that detail how failures are transformed and propagated from input to output (PAPADOPOULOS *et al.*, 2011). To support both techniques, compositional safety analysis tools have been proposed, such as Hierarchically Performed Hazard Origin and Propagation Studies (Hip-HOPS) (PAPADOPOULOS *et al.*, 2011; PAPADOPOULOS; MCDERMID, 1999), Error Model Annex, Version 2 (EMV2) specified in Architecture Analysis and Design Language (AADL) (DELANGE; FEILER, 2014), and CHESS (MAZZINI *et al.*, 2016b).

HiP-HOPS is a compositional safety analysis tool that takes a set of local component failure data, which describes how output failures of those components are generated from combinations of internal failure modes and deviations received at the components' inputs, and then fault tree analysis and FMEA tables synthesis that reflect the propagation of failures throughout the whole system. In turn, AADL EMV2 intends to automate safety analysis methods by supporting them through analyzable architecture fault models and uses the system and software architecture specified in AADL to define hazards, fault propagation, failure modes, and effects associated with components. Finally, CHESS is an open-source methodology and tool whose objective is to improve model-based practices to better address safety, reliability, and performance by guaranteeing the correctness of component development and composition for critical embedded systems. Besides such tools, meta-models have been proposed to support SoS design and safety analysis, such as Architecture for Multi-criticality Agile Dependable Evolutionary Open System-of-Systems (AMADEOS) (LOLLINI *et al.*, 2016) and Open Dependability Exchange (ODE) meta-model from Dependability Engineering Innovation for cyber-physical Systems (DEIS) project (WEI *et al.*, 2017). The ODE meta-model enables Cyber-Physical System (CPS) developers to capture various aspects of CPS, including architecture models, Hazard Analysis and Risk Assessment (HARA) models, failure logic models as well as assurance case models. In turn, AMADEOS is a common language allowing experts to collaborate on modeling, engineering, and analyzing SoSs. AMADEOS includes the SoS dependability package in which systems dependability attributes are modeled, such as systems safety, and safety data is modeled through the Unified Modelling Language (UML) concept of stereotype.

1.2 Motivation, Problem, and Justification

SoS differs from CSs due to its inherent characteristics: *autonomy* (the ability of a CS to make independent choices); *belonging* (CSs have the right and ability to decide if they belong or not to the SoS); *connectivity* (the ability to stay connected to other CSs and potential additions of

new CSs to the SoS); *diversity* (the SoS should be composed by CSs with visible heterogeneity); and *emergence* (formation of new properties as a result of developmental or evolutionary process, since the SoS development and operation are evolutionary) (BOARDMAN; SAUSER, 2006). Such characteristics and the desire for dynamic reconfiguration present severe difficulties for the traditional safety analysis approach. SoS characteristics make conventional safety analysis impractical because it is almost impossible to exhaustively enumerate all the possible interactions that might take place in an SoS of any considerable complexity. In this scenario, understanding the SoS failure modes can serve as a basis for improved identification and understanding of hazards at the SoS-level to avoid accidents, and mishaps not covered in safety analysis at the CS level (ALEXANDER; HALL-MAY; KELLY, 2004).

Unlike single systems, it is not yet clear how to analyze hazards and provide the required information for SoSs (BAUMGART; FRÖBERG; PUNNEKKAT, 2020). One reason is the system boundary. In single systems, the boundary is well-defined, and the components under the system can be enumerated. In SoSs, it is the opposite because the boundary can vary over time as part of a regular operation (e.g., a new vehicle can join the vehicle platooning system at any time during its operation) or an evolutionary development (e.g., a new type of truck is allowed to join the vehicle platooning system). Traditional safety analysis techniques are unsuitable for SoSs because they deal with only one failure at a time, and coincident failures are rarely considered (ALEXANDER; KELLY, 2006). Besides, safety-critical SoSs are complex systems with many failure modes and thus challenging to perform manual analysis (ALEXANDER; HALL-MAY; KELLY, 2004). Therefore, computational techniques are recommended to support the analysis.

As discussed, SoSs have inherent characteristics that differ from single systems, such as defined by Boardman and Sauser (2006), mainly regarding the boundary. Although existing compositional techniques and tools support safety analysis for complex systems, none of them considers SoS characteristics. Moreover, existent compositional meta-models have their limitations regarding SoS safety analysis. The ODE meta-model focuses on CPS, and it does not consider inherent characteristics of hazards at the SoS-level, such as the taxonomy to document and classify SoS hazards defined by (REDMOND, 2007). Finally, the AMADEOS meta-model focuses on SoS design modeling, not on modeling and analyzing SoS safety.

Therefore, this creates a need for safety approaches applicable for analyzing SoS (SABERI *et al.*, 2018). Novel safety analysis approaches at the SoS level are justified since automated safety analysis may benefit the industry to better execute the safety analysis on SoSs and, consequently, lead to safer SoSs (PAIGE *et al.*, 2008). In this context, existent compositional safety analysis approaches can be extended to the SoS-level, considering that SoS failures can be composed of the characterizations of individual CSs. This way flows of failure modes between CSs are identified and analyzed to define the SoS failure model (LISAGOR; MCDERMID; PUMFREY, 2006). This failure model can automate the generation the artifacts to perform traditional safety analysis, such as fault trees and FMEA tables, and hence help the SoS development

team to assure the safety of the whole system.

1.3 Objective

This work intends to answer the following Research Questions (RQs):

RQ$_1$ What are the models, methodologies, techniques, and tools to support safety analysis in SoS-based architectures available in the literature?

For answering RQ$_1$, this work presents the background concepts needed to understand the subject of this research, and a systematic mapping is performed to understand risk management practices in the context of SoS deeply. With this, the SoS safety analysis gaps can raise the research proposal.

RQ$_2$ How to support the design and safety analysis of SoSs?

For answering RQ$_2$, a meta-model to support SoS design and safety analysis is proposed, which consists of a structured way to model the information regarding SoS and its CSs to perform systems safety analysis. The SoS Safety Analysis (SoSSafe) meta-model has been built upon and extends the ODE meta-model from the DEIS Project. This work proposes extending the ODE meta-model by adding SoS concepts, hence considering SoS characteristics needed to perform SoS safety analysis.

RQ$_3$ How to conduct SoS safety analysis through compositional approaches?

For answering RQ$_3$, this work proposes a compositional approach to support SoS safety analysis named SoSSafe. SoSSafe is based on safety life-cycles widely used in the industry through safety standards, such as that proposed by International Organization for Standardization (ISO) 26262 (ISO, 2011) from the automotive domain and the Guideline for Development of Civil Aircraft and Systems ED-79A/SAE ARP 4754A (EUROCAE, 2010) from the aerospace domain, as well as compositional techniques. This approach encompasses the scope of SoS safety analysis, identifying combinations among system failure events that may lead the SoS to fail and classifying their risks to the overall SoS safety. The safety assets generated at the end of the approach are the synthesis of fault trees to analyze how system failures propagate throughout the CSs and FMEA tables to identify the most critical components. An illustrative case from the automotive domain has been proposed to demonstrate the applicability of SoSSafe.

1.4 Organization

The remainder of this document is organized into the following chapters:

Chapter 2 – Background: It presents the SoS, Systems Safety, and other concepts necessary for the reader to understand this work;

Chapter 3 – Systematic Mapping: It presents the results of a systematic mapping on risk management for SoS conducted to support this work;

Chapter 4 – SoSSafe Meta-model: It presents a structured way to model the information regarding SoS and its CSs to perform systems safety analysis;

Chapter 5 – SoSSafe: Compositional Systems-of-Systems Safety Analysis: It presents a compositional approach to support SoS safety analysis;

Chapter 6 – SoSSafe Illustrative Case: It presents the validation process of the proposed compositional approach;

Chapter 7 – Conclusions and Future Work: It presents a summary of the work contributions, as well as their limitations and future directions.

CHAPTER

2

BACKGROUND

2.1 Initial Considerations

To answer part of RQ_1, this chapter provides the background concepts needed for the reader to understand the subject of this research. In this context, Section 2.2 presents SoS concepts and definitions. Section 2.3 presents an overview of the safety analysis process. Finally Section 2.4 presents the final considerations.

2.2 System-of-Systems (SoS)

There is no precise and widely accepted characterization of SoS as an emerging field. The literature is diverse, and many attempts have defined and characterized SoS. In the SoS literature, the definition of SoS exposed by Maier (1998) has been widely cited, which considers SoS as a set of CSs with operational and managerial independence that cooperate to achieve a goal through emerging behavior that cannot be achieved considering each CS in isolation (NIELSEN *et al.*, 2015). In this context, the CSs have operational independence "if the SoS is disassembled into its CSs, the CSs must be able to operate independently usefully. That is, the components fulfill customer-operator purposes on their own". On the other hand, the CSs have managerial independence if "the CSs not only can operate independently, they do operate independently. The CSs are separately acquired and integrated but maintain a continuing operational existence independent of the SoS" (MAIER, 1998).

One of the main attributes of an SoS is its composition by CSs, in which each CS that composes the SoS may be situated in different geographical locations around the globe (LOPEZ, 2006; WOJCIK; HOFFMAN, 2006). CSs are standalone systems since there are independent development, management goals, and resources. Thus, CSs can provide one or more capabilities to one or more SoSs, in which *capability* can be defined as "the ability to achieve overall user objectives in a mission or business context" (ISO, 2019).

The concepts of SoS have been applied in a variety of domains, e.g., intelligent transportation (MATHEW, 2020), aerospace (GUARINIELLO *et al.*, 2019), energy (GUNAWAN *et al.*, 2017), healthcare (LEITE; SCHNEIDER; ADLER, 2018), military (CHEN; UNEWISSE, 2017), smart cities (ELSHENAWY; ABDULHAI; EL-DARIEBY, 2018), etc. In the current literature, it is possible to find several definitions focused on classifying and characterizing SoS (BOURQUE; FAIRLEY, 2014; JAMSHIDI, 2008). The main SoS characteristics are presented in Section 2.2.1 and an SoS classification is presented in Section 2.2.2.

2.2.1 SoS Characteristics

SoS characteristics are important because they can help to recognize or realize an SoS. Thus, some researchers have proposed distinct SoS definitions over time by focusing on distinguishing characteristics rather than presenting a single abstract description of it (GANDHI; GOROD; SAUSER, 2011). Of all, the most cited in the literature are proposed by Maier (1998), Boardman and Sauser (2006).

The taxonomy proposed by Maier (1998) proposes defining an SoS through the following characteristics:

Operational independence: The CSs of an SoS are independent and can operate in a useful way without depending on the SoS;

Managerial independence: Even collaborating with other CSs, CSs are self-governing and individually managed;

Geographical distribution: CSs are distributed over a large geographical area to exchange data with each other through communication networks;

Evolutionary development: The development of an SoS is evolutionary because the objectives and functionality can constantly be changing. In this way, CSs and capabilities can be added, modified, or removed anytime;

Emergent behavior: Through the collaboration between CSs, the objectives of an SoS cannot be achieved by or assigned to any of the CSs.

In turn, the taxonomy considered in this work has been proposed by Boardman and Sauser (2006). It is based on a review of over 40 definitions of SoS from the literature. In this taxonomy, an SoS may be classified into five characteristics: autonomy, belonging, connectivity, diversity, and emergence. A description of each characteristic is exposed as follows (BOARDMAN; SAUSER, 2006; GOROD; SAUSER; BOARDMAN, 2008):

Autonomy: The ability of a CS to make independent choices. Such independence can be both operational and managerial. Thus, a CS is free to reach its purpose. Any CS may fail to fulfill its purpose, but not because of autonomy;

Belonging: CSs have the right and ability to decide if they belong or not to SoS. Such choice is based on necessities, beliefs, or fulfillment of each CS;

Connectivity: The ability to stay connected to other CSs and potential additions of new CSs to the SoS. Therefore, CSs should be autonomous to make real-time connections to achieve and sustain their capabilities;

Diversity: The SoS should be composed of CSs with visible heterogeneity. By necessity, an SoS must be quite diverse in its capabilities compared to the limited functionality of a CS since the design of an SoS and its CSs are distinct. While CSs have defined and static scope, SoS must be open to diversity because of its uncertainty, persistent surprise, and disruptive innovation;

Emergence: SoSs can form new properties due to a developmental or evolutionary process since the development and operation of the SoS are evolutionary. Therefore, functions and capabilities are added, removed, or modified according to SoS needs.

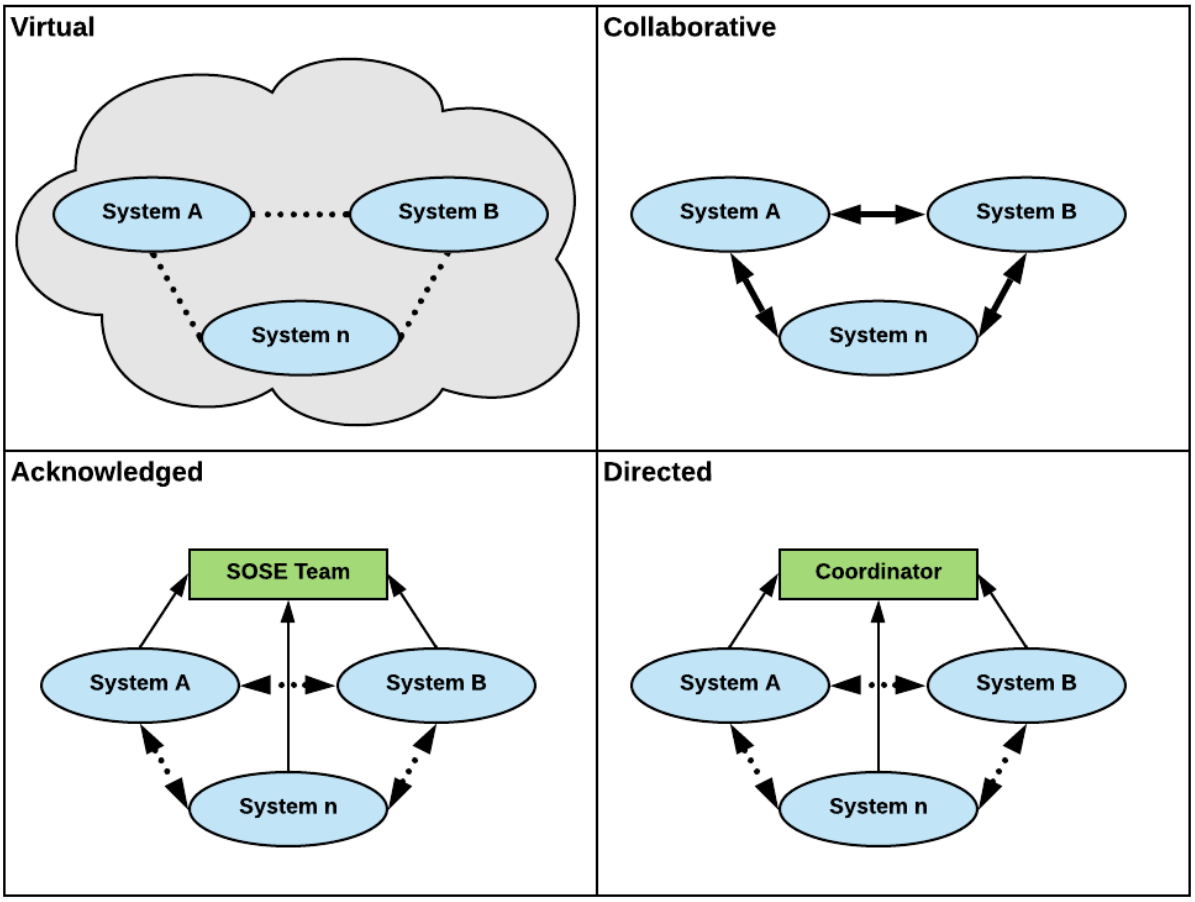
2.2.2 SoS Classification

In the current literature, there are several kinds of SoS classification. The most known classification describes different types of SoS based on the relationships among the CSs in the SoS. Thus, the SoS can be classified into four types as follows (MAIER, 1998; DAHMANN; BALDWIN, 2008; DAHMANN; JR; LANE, 2008):

Virtual: In Virtual SoS, there is no central management authority and a centrally agreed upon objective for the SoS. Virtual SoS must entrust with invisible mechanisms to maintain it (see Figure 1). National economies have been seen as Virtual SoSs since there are conscious attempts to architect these SoSs through politics. However, the long-term nature is determined by highly distributed, partially invisible mechanisms because the purposes expressed by the SoS emerge only through the collective actions of the CSs;

Collaborative: In Collaborative SoS, the central management organization does not have coercive power to run the system. Thus, the systems must, more or less, voluntarily collaborate to fulfill the agreed-upon central objectives (see Figure 1). The Internet is an example of Collaborative SoS because the Internet Engineering Task Force works out standards but has no power to enforce them. The physical elements of the Internet, such as routers, backbones, and nodes, are operational and managerial independent systems, as they operate

Figure 1 – SoS characteristics.



Source: Adapted from Lane (2013).

primarily for the benefit of each physical element. However, such elements interact with each other to yield emergent properties of large scale and complexity;

Acknowledged: Acknowledged SoS has recognized objectives, a designated manager, and resources for the SoS. However, the systems keep their independent ownership, objectives, funding, development, and maintenance approaches. Thus, changes in the SoS are based on cooperative agreements between the System-of-Systems Engineering (SoSE) Team and the CSs (see Figure 1). Acknowledged SoS can be used in commercial products or services in user-specific applications. This type of SoS has particular objectives the majority may not fully support by the commercial product or system, which has its own objectives and development path. Therefore, this situation characterizes what is being faced by a variety of organizations today as they cope with addressing changes in their business objectives and environment, including relationships, but without the time or resources to replace their systems supporting persistent aspects of their business but now in a new context;

Directed: Directed SoS is a type of SoS built and managed to perform specific objectives. The coordinator centrally defines its management to fulfill the established objectives or any

new ones the system owners may wish to address. The systems can operate independently, but their operational mode is subordinated to the SoS objectives (see Figure 1). An example of Acknowledged SoS is the integrated air defense networks because they are centrally managed to defend a region against enemy systems, although its CSs can operate independently.

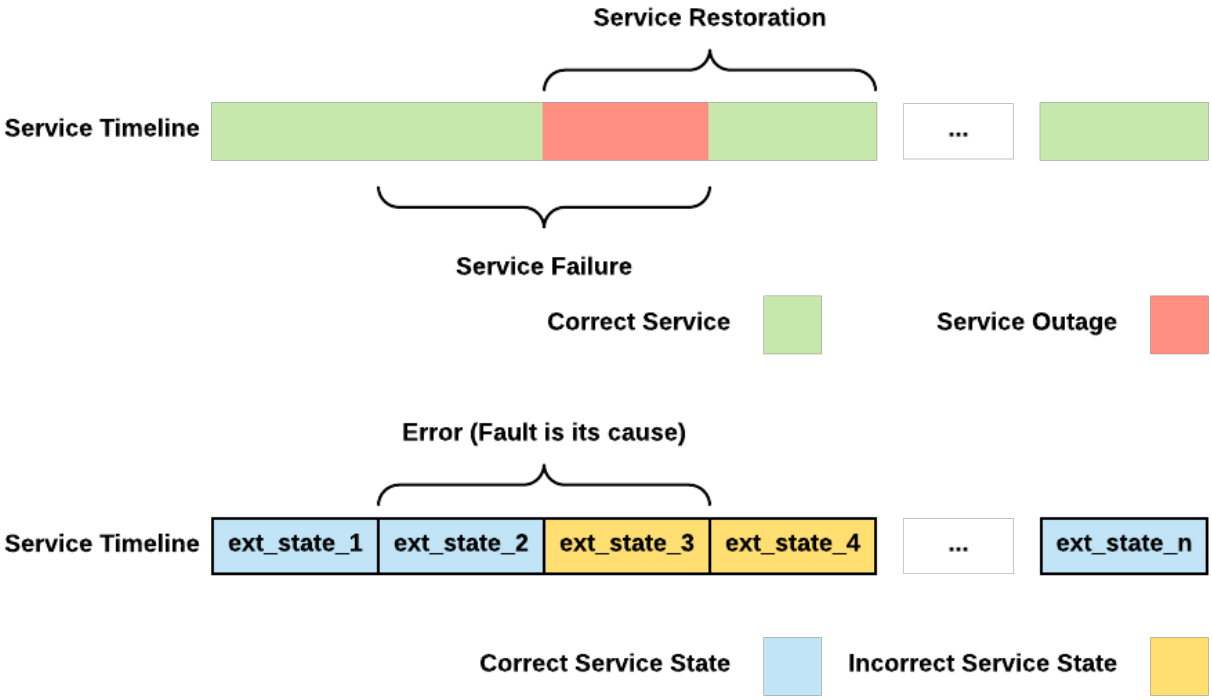
2.3 Safety Analysis

This section provides the background for the reader to understand SoS safety analysis. Section 2.3.1 presents the safety terminology adopted in this document. Section 2.3.2 presents the generic safety life-cycle. Section 2.3.3 presents HARA. Finally, Section 2.3.4 presents the safety analysis techniques and tools.

2.3.1 Safety Terminology

The following terms related to safety used in this document are explained to support the reader in understanding this work.

Figure 2 – Differences between the terms failure, error, and fault.

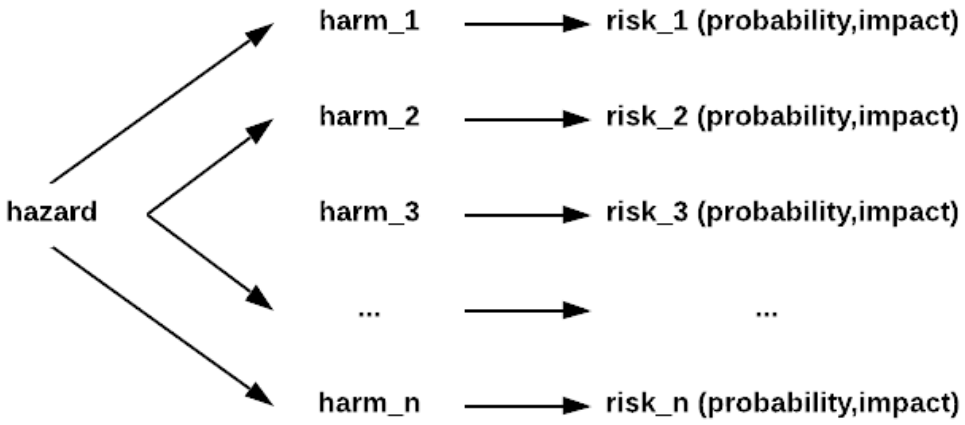


Source: Elaborated by the author.

2.3.1.1 Error, Fault, Failure and Hazard-related Concepts

A **system** is an entity that communicates with other entities. Examples of entities are other systems, such as hardware, software, humans, and the physical world with its natural

Figure 3 – Relationship between the terms hazard, harm, and risk.



Source: Elaborated by the author.

phenomena. As shown in Figure 2, a system performs a system function throughout its operation, namely **service**. A **correct service** is delivered when the service implements the expected system function, whereas a **service outage** is the opposite since it implements an incorrect system function. In this context, a **failure** is "an event that occurs when the delivered service deviates from correct service". In other words, it is a transition from a correct service to a service outage. An example of failure is when the airbag does not work during a strong car collision. The opposite of the failure, i.e., the transition from incorrect service to correct service, is a **service restoration** (AVIZIENIS *et al.*, 2004a). A failure may assume different forms that are called **failure modes** (AVIZIENIS *et al.*, 2004a).

As depicted in Figure 2, a service assumes a sequence of the system's external states during its operation. An **error** is the part of the entire state of the system that may lead to its subsequent failure, and its adjudged or hypothesized cause is called a fault. Some errors do not reach the system's external state and cause failure. A fault is **active** when it causes an error, otherwise it is **dormant** (AVIZIENIS *et al.*, 2004a). An example of fault is a buffer overflow, which occurs when a system tries to store more data in a fixed block of memory than the allocated block supports.

Depending on how the system fails, its failures can cause harm which is a physical injury or damage to the health of persons. A **hazard** is "a real or potential condition that can cause human injury, damage to health, property, or the environment, or damage to or loss of a system". When a hazard is combined with an **operational situation** ("scenario that can occur during a vehicle's life", (e.g., driving or parking), a **hazardous event** occurs. A **risk** is associated with harm, which is the combination of the probability of occurrence and impact of harm. Analyzing Figure 3, a hazard may be the source of one or more harms. One or more hazards may contribute to a catastrophic incident or accident. A risk is **unreasonable** or **unacceptable** when it is "judged to be unacceptable in a certain context according to valid societal moral concepts" (ISO, 2011).

2.3.1.2 Dependability

Dependability of a system is "the ability to avoid service failures that are more frequent and more severe than is acceptable". Dependability is an integrating concept that wraps the following properties (AVIZIENIS *et al.*, 2004a):

Availability: Readiness for correct service;

Reliability Continuity of correct service;

Safety: Absence of catastrophic consequences on the user(s) and the environment;

Integrity: Absence of improper system alterations;

Maintainability: Ability to undergo modifications and repairs.

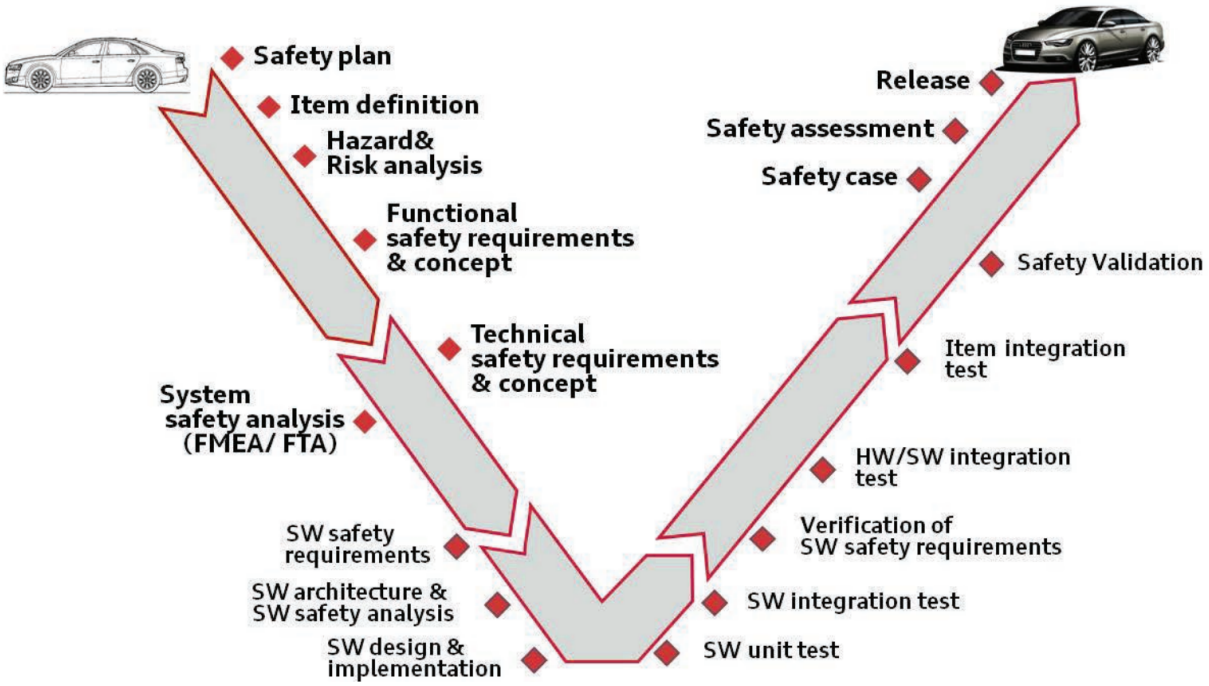
In this work, the focus is on the safety attribute. Regarding risk assessment, **safety** means "absence from unacceptable risk" (ISO, 2011). The concept of safety is similar to the concept of **reliability**, which is defined as "the probability that a system will perform its intended function for a specified time under a set of specified environmental conditions". However, the definitions of safety and reliability are distinct because reliability is concerned with keeping the system away from all types of failures, whereas safety is only concerned with hazardous failures (LEVESON, 1986).

Safety requirements is defined as the necessary risk reduction measures associated with a given hazard or failure component (MOD, 2007). There are three types of safety requirements: (i) **System Safety Requirements**, which is "the risk reduction measure to mitigate hazard effects" (MOD, 2007); (ii) **Functional (or derived) Safety Requirement** is the "specification of implementation-independent safety behaviour, or implementation-independent safety measure, including its safety-related attributes" (ISO, 2011); and (iii) **Safety Integrity Requirement**, which represents the reliability, e.g., in terms of probability and severity attributes, associated with a given component failure/fault, hazard, or functional safety requirement" (OLIVEIRA *et al.*, 2019). The artifacts produced throughout the safety life-cycle are called *safety assets*, which include safety requirements as well as a safety plan and HARA report (ISO, 2011).

2.3.2 Safety Life-cycle

Standards establish that the dependability properties of a critical system should be analyzed and demonstrated at different levels of abstraction before its release for operation, as illustrated in Figure 4. During hazard analysis, the potential threats and risks posed to overall system safety are identified and determined at the requirements level. At the design level, the propagation of system failures throughout architectural subsystems should be analyzed during component fault modeling, using Failure Logic Analysis (FLA) or FTA techniques. Finally,

Figure 4 – V-model with safety-related work products.



Source: Käßmeyer, Schulze and Schurius (2015).

at the component level, it is necessary to identify how components can contribute, directly or indirectly, to the occurrence of system failures (named hazards) using FMEA or Failure Modes, Effects, and Criticality Analysis (FMECA) techniques.

Several safety standards have been proposed to cover a range of domains, such as aerospace and automotive. In the aerospace domain, there is the Guideline for Development of Civil Aircraft and Systems ED-79A/SAE ARP 4754A from the European Organisation for Civil Aviation Equipment (EUROCAE) which covers the development of aircraft systems taking into account the overall aircraft operating environment and functions. ARP 4754A includes validation of requirements and verification of the design implementation for certification and product assurance. EUROCAE ED-79A/SAE ARP 4754A guides demonstrating compliance with regulations, supporting companies in developing and meeting their internal standards by considering standard guidelines (EUROCAE, 2010). In turn, ISO 26262-1 Road vehicles — Functional Safety applies in the automotive domain to safety-related systems that include one or more Electrical and Electronic (E/E) systems and that are installed in series production passenger cars with a maximum gross vehicle mass up to 3,500 kilogram (kg). ISO 26262-1 does not address unique E/E systems in special purpose vehicles, such as vehicles designed for drivers with disabilities (ISO, 2011). HARA has preliminary activities of safety life-cycle in both safety standards, as illustrated in Figure 4, which is denominated *Hazard & Risk Analysis*. HARA aims to identify hazardous functional failure conditions, determine the risks associated with each identified hazard, and establish safety requirements to minimize or eliminate hazard effects. In

ISO 26262-1, this activity is named HARA, whereas in EUROCAE ED-79A/SAE ARP 4754A, this activity is named Functional Hazard Assessment (FHA) in both aircraft and system levels (EUROCAE, 2010; ISO, 2011). The sub-activities of HARA and FHA are further detailed in the following.

2.3.3 Hazard Analysis and Risk Assessment (HARA)

In the context of safety standards, there is no consensus on a universal approach. Thus, safety standards may vary according to industry, domain, or region (MOD, 2007). However, most safety standards cover basic engineering activities, such as Hazard and Risk Assessment. As described earlier, each safety standard has a specific name for this activity (e.g., HARA and FHA). However, both have similar sub-activities as follows (OLIVEIRA, 2016):

1. **Hazard Identification:** In this sub-activity, hazardous functional failure conditions that can lead the system to an unsafe state are identified. First, it is necessary to specify and understand the system and its target operating environment to define the scope of the analysis. Hazard Identification can be performed through checklists, "what-IF" analysis, Functional Failure Assessment (SAE, 1996), or HAZard and OPerability Studies (HAZOP) (DUNJÓ *et al.*, 2010). Hazards are identified iteratively, and then the information of hazards identified is updated during the safety life-cycle in order to improve the knowledge about the system hazards (HABLI, 2009);

2. **Risk Assessment:** It relates to the classification of the risk posed by each identified hazard. Standards use different attributes for classifying the risk posed by system hazards. In ISO 26262-1, the risk of a hazard is classified based on: (i) Severity (how severe a potential accident is); (ii) Probability of Exposure (how often this situation occurs); and (iii) Controllability (if the critical situation can be controlled by an operator or educated personal). Through these parameters, each hazardous event shall determine an Automotive Safety Integrity Level (ASIL). ASIL refers to an abstract classification of inherent safety risk in an automotive system or elements of such a system (ISO, 2011). In EUROCAE ED-79A/SAE ARP 4754A, each Failure Condition is classified based on its severity (i.e., Catastrophic, Hazardous/Severe-Major, Major, Minor, or No Safety Effect) and probability of occurrence (EUROCAE, 2010);

3. **Allocation of Safety Requirements:** Finally, from the analysis of the outputs provided by the HARA, safety requirements are assigned to mitigate hazards and component failure effects. Safety requirements can be stated as functional safety or safety integrity requirements. In ISO 26262-1, safety goals "are top-level safety requirements assigned for an item in the form of ASIL, and a safety goal shall be defined for each hazardous event. Safety goals define the safety requirements necessary to avoid an unreasonable risk for each hazardous event (ISO, 2011). In EUROCAE ED-79A/SAE ARP 4754A, a set

of Development Assurance Level (DAL) are assigned to mitigate the effects of failure conditions on the overall safety (EUROCAE, 2010).

Whereas HARA has been used in the automotive domain, FHA is used in the aircraft domain. Still, the activity performed by FHA is similar to HARA (ISO, 2011; EUROCAE, 2010). The main difference between HARA and FHA is that HARA is performed only at one level (vehicle), while FHA is performed at two levels (systems and aircraft) (ISO, 2011; EUROCAE, 2010).

Regarding SoS hazards, the study proposed by Baumgart, Fröberg and Punnekkat (2017) presents a structured process for identifying potential hazards in SoS, exemplified in a quarry site automation context. To organize the hazards identified by the proposed process, Redmond taxonomy is used (REDMOND, 2007). Such taxonomy classifies SoS hazards through the five categories:

Interface Hazards: These hazards are caused by sharing faulty data through a defined channel. The hazard occurs in the receiving system, which relies on the received data. It is necessary to define which channels shall be used and which kind of data shall be exchanged between CSs in SoS;

Proximity Hazards: It may occur if different CSs operate within a short physical distance. A failure in one CS may lead to hazards in the other CS. Characteristics of the site may cause unintended behavior, such as electromagnetic compatibility interference from high-voltage charging equipment;

Resource Hazards: They result from insufficient shared resources or resource conflicts. Limited shared communication channels or limited electric charging possibilities are examples of unintended behavior of the SoS;

Reconfiguration Hazards: They are related to changes in the configuration of the SoS, which may depend on operational demands or replacement of CSs;

Interoperability Hazards: They result from interpreting the information from one CS by a second CS in a way that the first CS did not intend. If, for example, an autonomous system is sending a status of not being shut down properly and the site management CS misunderstands this status, approval for safe entering the autonomous operating area might be provided.

The following section presents safety analysis techniques and tools to support hazard identification and analysis, as well as the identification of its causes.

2.3.4 Safety Analysis Techniques and Tools

A set of techniques available in the literature can be used to support safety life-cycle activities. These techniques are classified into traditional and model-based. Model-based techniques are split into two categories: compositional and extensions of formal verification techniques. In this section, it is presented an overview of both traditional (Section 2.3.4.1) and model-based safety analysis techniques (Section 2.3.4.2).

2.3.4.1 Traditional Safety Analysis Techniques

The following presents an overview of traditional safety analysis techniques, namely HAZOP, FTA, and FMEA. These techniques are used to identify potential failures in a system so that the generated information can be used to correct or prevent the identified faults. HAZOP (Section 2.3.4.1.1), FTA (Section 2.3.4.1.2), and FMEA (Section 2.3.4.1.3) are well-established safety analysis methods primarily used in the engineering of safety-critical systems in aerospace, automotive, railway, and nuclear power plant domains.

2.3.4.1.1 HAZard and OPerability Studies (HAZOP)

HAZOP study is a structured and semi-formalized team-based procedure focused on the study of a system under design to identify and evaluate potential hazards that may constitute risks to personnel or equipment or prevent the efficient operation of the system (DARAMOLA *et al.*, 2011). The first application of HAZOP was in the chemical industry (KLETZ, 1997). Over time, HAZOP has been applied in other areas, such as avionics systems and computer-assisted braking systems, being valuable and essential for the safety assurance of computer-based systems (MCDERMID *et al.*, 1995).

HAZOP relies on guide words, as shown in Table 1, which are combined with process parameters (e.g., *value* and *flow*) in order to reveal deviations from normal process intent or operation. For example, applying the guide words 'before' and 'after' to the timing of a message in a sequence diagram can instigate designers to consider whether or not hazards can occur if the message is sent earlier or later within the sequence of messages (HANSEN; WELLS; MAIER, 2004). After determining the deviations, experts explore each deviation's possible causes and consequences to generate pairs of cause-consequence. For each pair, safeguards are identified whose purpose is to prevent, detect, control, or mitigate the hazardous situation (DUNJÓ *et al.*, 2010; WINTHER; JOHNSEN; GRAN, 2001). HAZOP studies are based on examining design representations of a system. The recommended steps in a HAZOP study are (HANSEN; WELLS; MAIER, 2004):

1. Identify each system in the design representation;
2. Identify attributes, i.e., physical or logical properties, for each system;

3. Investigate deviations from design intent by applying guide words to attributes;

4. Investigate, for each deviation, the causes and consequences.

Table 1 – Generic guide words for programmable electronics.

Guide Word	**Interpretation**
No	This is the complete negation of the design intention. No part of the intention is achieved, and nothing else happens.
More	This is a quantitative increase.
Less	This is a quantitative decrease.
As well as	All the design intention is achieved together with additions.
Part of	Only some of the design intention is achieved.
Reverse	The logical opposite of the intention is achieved.
Other than	Complete substitution, where no part of the original intention is achieved, but something quite different happens
Early	Something happens earlier than expected relative to clock time
Late	Something happens later than expected relative to clock time.
Before	Something happens before it is expected, relating to order or sequence.
After	Something happens after it is expected, relating to order or sequence

Source: Ministry of Defence (2000).

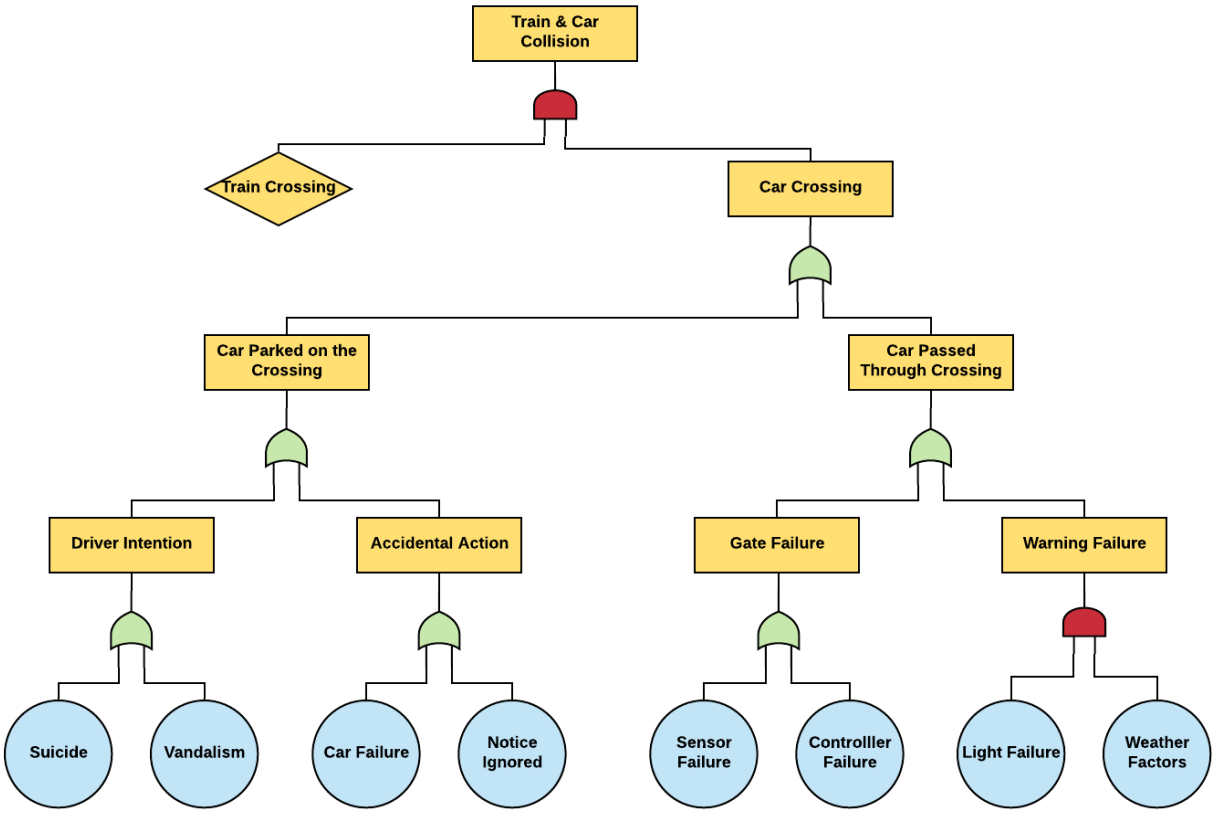
2.3.4.1.2 Fault Tree Analysis (FTA)

FTA (VESELY *et al.*, 2002) is a deductive technique that considers the analysis of a top-event, typically a system-level failure identified during hazard analysis, and then deduces its causes. A top event of a fault tree can be an identified hazard using functional failure assessment or HAZOP. Fault tree analysis is performed based on the preliminary design of the system architecture to identify failures on components named basic events, leading to the occurrence of the top-level event.

FTA can be performed qualitatively via logical analysis of the relations between the top and basic events or quantitatively via probabilistic analysis. The qualitative analysis considers basic events, their relationships, and contributions to the occurrence of a top event. The quantitative analysis calculates the probability of occurrence of a given top event based on the analysis of the probability of occurrence of each basic event.

A fault tree provides a graphical representation of combinations of component failures that contribute to the occurrence of the top event connected to one or more basic events via

Figure 5 – Example of fault tree for the railway level crossing.



Source: Thapaliya and Kwon (2017).

logical gates, such as AND, OR, or NOT. Figure 5 illustrates an example of a fault tree for the railway level crossing. The collision between car and train top-event is represented in level 0. Analyzing the fault tree, the top event can be triggered if both car and train cross the railway. Analyzing a top-down way makes it possible to determine the failure caused by leaf nodes.

FTA can be used to support the Preliminary System Safety Analysis (PSSA) phase from EUROCAE ED-79A/SAE ARP 4754A safety process (EUROCAE, 2010) and ISO 26262-1 Safety Validation. Fault tree analysis allows refining high-level safety requirements assigned to system hazards during HARA to derive safety requirements to be assigned to system components, thus supporting the decomposition of safety requirements required in both automotive ISO 26262-1 Functional Safety Concept and EUROCAE ED-79A/SAE ARP 4754A PSSA.

2.3.4.1.3 Failure Modes and Effects Analysis (FMEA)

Unlike FTA, FMEA is an inductive technique that infers the potential effects of knowing component failures on the overall system safety (PUMFREY, 1999). Its objectives are providing a systematic examination of possible system single failure, analyzing the effects of each failure mode on system operation, recording the causes of the failure modes, and specifying appropriate detection procedures and corrective actions for each failure mode (SEVCIK, 1981). As shown in

Table 2 – Example of FMEA table.

Component failure	**Direct effects on the system**	**Effects caused in conjuntion with other events**
C1	F1	–
C2	F1	–
C3	–	F1 (C4)
C4	–	F1 (C3)
C5	F1, F2	–
C6	–	F1, F2 (C7)
C7	–	F1, F2 (C6)
C8	F2	–
C9	F2	–

Source: Papadopoulos (2013).

Table 2, the results obtained by FMEA can be represented in the form of a table listing the effects of each component failure on the rest of the system, in which F1 and F2 are system failures, and C1—C9 are component failures. There are no direct effects on the system for C3, C4, C6, and C7. However, they have further effects as, for example, C3 and C4 both occurring in conjunction will cause F1.

FMEA can be extended through a Criticality Analysis (CA) procedure, in which failure modes are rated by assessing their criticalities. The combination between FMEA and CA procedures corresponds to FMECA (BOUTI; KADI, 1994). The first step of FMECA is to identify all possible potential failure modes of the system through systematic brainstorming. After that, criticality analysis is performed on each potential failure mode taking into account the risk factors (LIU; LIU; LIU, 2013):

- **Occurrence (O):** the probability of the failure;
- **Severity (S):** the severity of the failure;
- **Detection (D):** how easy it is to detect the failure.

In order to prioritize the failure modes for corrective actions, the Risk Priority Number (RPN) is calculated for each failure mode. First of all, each risk factor is evaluated using a 10-point scale, where each risk factor has its scale. In Table 3, it is shown an example of a 10-point scale for the risk factor *Occurrence*. In this way, the RPN is calculated through the equation RPN=O.S.D. The higher the RPN of a failure mode, the greater the risk is for system

safety. For example, a failure mode x with values $O = 5$, $S = 4$ and $D = 3$ has a higher priority than a failure mode y with values $O = 2$, $S = 3$ and $D = 5$ because $RPN_x = 60 > RPN_y = 30$.

After applying corrective actions, all RPNs should be calculated again to check if the risks have gone down and verify the efficiency of the corrections for each failure mode (LIU; LIU; LIU, 2013).

Table 3 – example of ratings for the occurrence of a failure mode.

Probability of failure	**Possible failure rates**	**Rank**
Extremely high: failure almost inevitable	$\geq$ in 2	10
Very high	1 in 3	9
Repeated failures	1 in 8	8
High	1 in 20	7
Moderately high	1 in 80	6
Moderate	1 in 400	5
Relatively low	1 in 2000	4
Low	1 in 15,000	3
Remote	1 in 150,000	2
Nearly impossible	$\leq$1 in 1,500,000	1

Source: Liu, Liu and Liu (2013).

Although there are similarities in the structure and analysis process of component FMEA/FMECA and HAZOP, the latter is a predictive technique commonly used early on the safety life-cycle to identify potential threats to system safety and assign safety requirements. On the other hand, FMEA and FMECA are used later in the safety life-cycle to demonstrate that the system addressed the safety requirements (PUMFREY, 1999). FMEA technique supports EUROCAE ED-79A/SAE ARP 4754A System Safety Analysis process, and automotive ISO 26262-1 Functional Safety Analysis activity (ISO, 2011).

HAZOP, FTA, and FMEA/FMECA provide valuable information about the system's faulty behavior, complementing each other. HAZOP provides information about the potential system failures that can arise from the system architecture, their risks, and assigned safety requirements. FTAs provide the causal information for system failures, and FMEA provides a view of the impact of component failures on overall safety. However, both techniques are primarily manual methods. Applying these methods to larger and complex systems can be a laborious and error-prone task (PAPADOPOULOS *et al.*, 2011).

Researchers and aircraft manufacturers have recognized the limitations of traditional safety analysis techniques with the revision of the SAE ARP 4761 (COMMITTEE *et al.*, 1996)

document that includes a specific section dedicated to model-based automated safety analysis. Therefore, some tools and techniques have been developed to automate the system safety process. Such tools are exposed as follows.

2.3.4.2 Compositional Safety Analysis Techniques

Modern safety analysis techniques have been proposed to address the limitations of traditional techniques related to the lack of automated support. These techniques are categorized into compositional and formal verification approaches to support system safety analysis. Compositional approaches are built upon the failure logic model that characterizes the failure behaviors of individual components by connecting output failure modes of components to input failure modes of other components (LISAGOR; MCDERMID; PUMFREY, 2006). FPTN (Section 2.3.4.2.1) and FPTC (Section 2.3.4.2.2) are well-known safety analysis techniques used in an extensive range of applications. The main compositional safety analysis techniques are presented in Section 2.3.4.2.3.

2.3.4.2.1 Failure Propagation and Transformation Notation (FPTN)

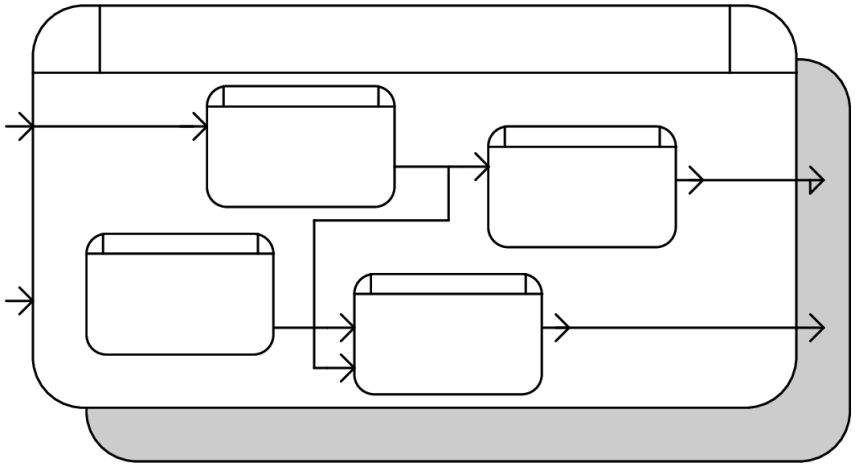
One of the earliest compositional safety analysis techniques is FPTN, which provides a graphical notation for expressing the failure behavior of systems with complex internal structures. FPTN is based on the idea of component modules that describe the generation and propagation of system component failures, which are connected via inputs and outputs to other modules to allow the combination and propagation of faults between system modules. These component modules can be composed into subsystems to build a system hierarchy (FENELON; MCDERMID, 1993). FPTN is a bridge between FTA and FMEA since the cause and effect of the failures can be studied (PAPADOPOULOS *et al.*, 2011). A deficiency of FPTN is that the system failure model is separate from the system architectural model, so any changes to the system model can lead to being out of sync with the system failure models (WALLACE, 2005). As illustrated in Figure 6, the system comprises modules connected via inputs and outputs to other modules, allowing the combination and propagation of failures between the modules. In Figure 7, modules have standardized attributes; a name, a criticality level, and (where applicable) an indication of any recovery mechanisms which might exist in that module (FENELON; MCDERMID, 1992).

2.3.4.2.2 Fault Propagation and Transformation Calculus (FPTC)

FPTC has been proposed to address the FPTN limitations. Unlike FPTN, FPTC combines the fault model with the system architectural model, in which any changes made in the architectural model do not require a new failure model to be built. Thus, only a part of the failure model needs updating.

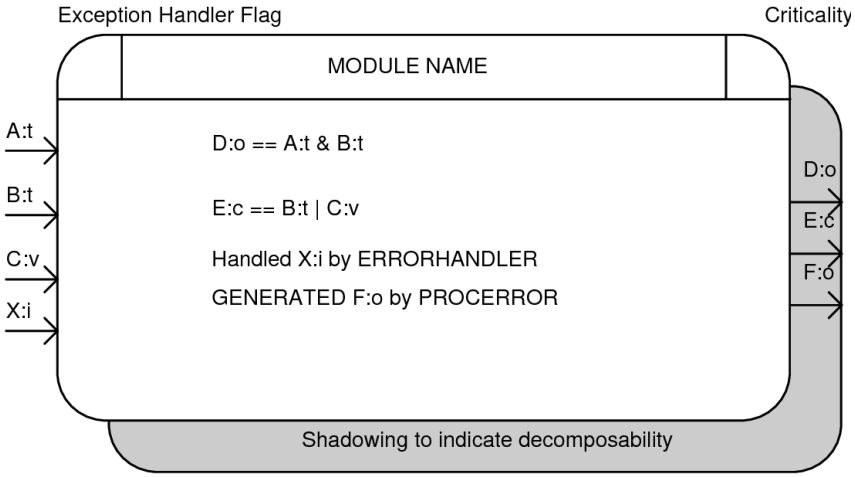
FPTC defines failure classes, such as omission, commission, and value failures, which are directly specified by annotations in the system model's components. Input and output ports

Figure 6 – FPTN combination of the modules.



Source: Fenelon and McDermid (1992).

Figure 7 – FPTN notation of each module.



Source: Fenelon and McDermid (1992).

from system components transmit fault information to other system components through logical expressions that detail the system's faulty behaviors. Therefore, such expressions and the system architectural model can be evaluated as a token-passing network, identifying the effects of each component failure on the system as a whole (WALLACE, 2005; PAPADOPOULOS *et al.*, 2011).

An extension to FPTC proposed by Ge, Paige and McDermid (2009) supports quantitative analysis by including probabilistic values for each failure expression. A disadvantage of FPTC over FPTN is that while FPTN supports deductive and inductive analysis, FPTC only supports inductive analysis. Thus, the information provided by an FTA is more challenging to obtain with FPTC. Moreover, systems with many failure modes are prone to combinatorial explosion (PAPADOPOULOS *et al.*, 2011).

2.3.4.2.3 Compositional Safety Analysis Tools and Meta-Models

In the current literature, tools are available to provide automated support for FPTN and FPTC compositional safety analysis techniques. Hip-HOPS (PAPADOPOULOS *et al.*, 2011), AADL EMV2 (DELANGE; FEILER, 2014), and CHESS (MAZZINI *et al.*, 2016b) are important tools and they are described as follows:

- **Hip-HOPS**: it is a toolset that supports the description of the failure logic of components and automatic synthesis of component failure data into fault trees that reflect the propagation of failures throughout the whole system architecture. From the synthesized fault trees, Hip-HOPS generates both quantitative and qualitative results into FMEA tables (PAPADOPOULOS *et al.*, 2011; PAPADOPOULOS; MCDERMID, 1999);
- **AADL EMV2**: this tool is an extension to the AADL standard, which aims to "automate safety analysis methods by supporting them through analyzable architecture fault models". It uses the system and software architecture specified in AADL to define hazards, fault propagation, failure modes, and effects associated with components. In order to facilitate incremental and scalable automated safety analysis, the compositional fault behavior specifications are defined as annotations in the system model (DELANGE; FEILER, 2014).
- **CHESS**: it is an open source methodology and tool whose objective is to improve model-based practices to address safety, reliability, and performance. It guarantees the correctness of component development and composition for critical embedded systems. It helps system architects interpret human, organizational, and technological entities through components and model their behaviors concerning erroneous and fault-tolerance (MAZZINI *et al.*, 2016b).

A common characteristic of Hip-HOPS, AADL EMV2, and CHESS is that the failure model is integrated into the system architectural model. However, these tools do not take into account the inherent SoS characteristics. Besides such tools, similar works present meta-models in the context of SoS dependability and safety analysis. ODE meta-model enables CPS developers to capture various aspects of CPS, including architecture, HARA, failure logic, and assurance case models. In turn, AMADEOS is a common language allowing experts to collaborate on modeling, engineering, and analyzing SoSs. AMADEOS includes the SoS dependability package in which systems dependability attributes are modeled, such as systems safety, and safety data is modeled through the UML concept of stereotype. Existent compositional meta-models have their limitations regarding SoS safety analysis. ODE meta-model focuses on CPS since it does not consider inherent characteristics of hazards at the SoS-level, such as the taxonomy to document and classify SoS hazards defined by (REDMOND, 2007). Finally, the AMADEOS meta-model focuses on SoS design modeling, not on modeling and analyzing SoS safety.

2.4 Final considerations

This chapter presented the concepts of SoS and safety analysis to provide the theoretical foundation for this research. To contribute to answering RQ_1, Chapter 3 presents a systematic mapping conducted in this work which intends to identify the state-of-the-art of SoS safety analysis, particularly SoS risk management techniques. Such systematic mapping is related to SoS risks and approaches and tools to manage the risks exposed in the SoS literature.

CHAPTER

3

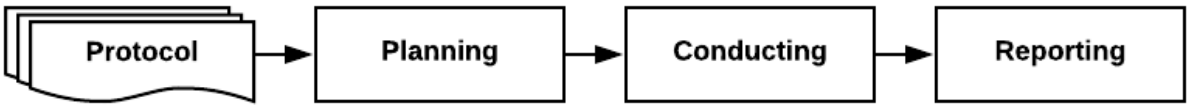
SYSTEMATIC MAPPING

3.1 Initial Considerations

To answer part of RQ_1, this chapter provides a systematic mapping to identify the state-of-the-art of SoS safety analysis with a focus on SoS risk management techniques. Systematic mapping is a broad review of primary studies on a specific topic to identify what evidence is available on the topic (KITCHENHAM; CHARTERS, 2007). This chapter presents a systematic mapping to identify existing primary studies related to risk management practices for SoS. This systematic mapping follows the process proposed by Kitchenham and Charters (2007) that comprises planning, conducting, and reporting phases, as shown in Figure 8. This systematic mapping identified SoS risks, SoS risk management approaches, tools, and their differences to system-level risk management approaches and tools.

The systematic mapping exposed as follows has also been described in a paper published at the 2020 IEEE International Conference on Software Architecture Companion (ICSA-C) (LOPES *et al.*, 2020).

Figure 8 – Systematic Mapping Process.



Source: Adapted from Kitchenham and Charters (2007).

The remainder of this chapter is organized as follows. Section 3.2 describes the planning phase. Section 3.3 presents the conducting phase. Section 3.4 presents the reporting phase. Section 3.5 presents the threats to validity. Finally, Section 3.6 presents the final considerations.

3.2 Planning

In this phase, the research objectives and the systematic mapping protocol were established. For this research, the protocol contains the following items:

- Research objectives and research questions;
- Search strategy;
- Selection criteria (i.e., inclusion and exclusion criteria);
- Quality Assessment;
- Data extraction and synthesis method.

3.2.1 Research Objectives and Research Questions

The main goal of this systematic mapping is to obtain state of the art on risk management practices and techniques for SoS focused on identifying the differences between risk management for SoS and CSs and the key SoS risks. From this objective, the following Systematic Mapping Research Question (SMRQ) were derived:

$SMRQ_1$: Which are the differences between risk management for SoS and risk management for CSs?

$SMRQ_2$: Which are the key SoS risks?

$SMRQ_{2.1}$: Which are the approaches used to manage risks at the SoS level, their strengths and weaknesses?

$SMRQ_{2.2}$: In which domains have these risks occurred?

3.2.2 Search Strategy

The following relevant keywords were identified from the analysis of the research questions to be further considered in search strings: *System of Systems* and *risk*. Combining these terms using logical operators (AND, OR, NOT), it was obtained the following general search string:

(“System of Systems” OR “Systems of Systems” OR “System-of-Systems” OR
“Systems-of-Systems” OR “SoS”)
AND
(“risk” OR “risks”)

The search string was validated by defining a control group (CONROW, 2005; HAIMES, 2012b; PINTO; MCSHANE; BOZKURT, 2012), with the support of an SoS expert. Through the primary studies of the control group, it was verified that the search string was properly defined to find primary studies relevant to this systematic mapping.

This search string was applied to identify primary studies in the following five databases: *ACM Digital Library*, *IEEE Xplore*, *ScienceDirect*, *Scopus* and *Web of Science*. The *Springer* database was not used because this database does not have the function of searching studies only by metadata, i.e., searching only by the title text, abstract, and keywords. The function of searching metadata is important to prevent irrelevant articles from being selected. The criteria established by Dieste and Padua (2007) were considered in selecting the databases for this systematic mapping.

3.2.3 Selection Criteria

The selection criteria were used to evaluate the studies obtained in each database to include the relevant primary studies to answer the research questions and exclude the primary studies that do not add knowledge to this systematic mapping. The Inclusion Criteria (IC) and Exlusion Criteria (EC) used were:

IC_1: Primary study that addresses risk management processes for SoS;

IC_2: Primary study that addresses the key risks or challenges for SoS;

EC_1: Primary study not written in English;

EC_2: Primary study that only addresses the risk management process for CSs;

EC_3: Primary study that only addresses the key risks or challenges for CSs;

EC_4: Primary study not available for online reading or download.

3.2.4 Quality Criteria

It was defined and used a checklist with seven Quality Criteria (QC), based on the quality assessment criteria proposed by Kitchenham and Charters (2007), to analyze the quality of the selected primary studies:

QC_1: Is there a rationale for why the primary study was undertaken?

QC_2: Is an overview of the state of the art of the area in which the primary study is developed presented?

QC_3: Is there an adequate description of the context in which the work was carried out?

QC_4: Is a clear justification for the methods used during the primary study provided?

QC_5: Is there a clear statement of the contributions and sufficient data to support them?

QC_6: Are the credibility and limitations of their findings explicitly discussed?

QC_7: Are the perspectives of future works discussed?

3.2.5 Data Extraction Form

In order to answer the SMRQs, it was defined a data extraction form to extract the following information from the identified primary studies:

F_1: Does the primary study address the risk management processes for SoS? If so, what are they?

F_2: Does the primary study take a comparative study between risk management for SoS and risk management for CSs? If so, what are they?

F_3: Does the primary study discuss the advantages or disadvantages of managing risks for SoS? If so, what are they?

F_4: Does the primary study propose any approach to manage risks for SoS? If so, describe the approach advantages, difficulties, limitations, or challenges;

F_5: What is the application domain targeted by the primary study?

F_6: Does the primary study mention risks that may arise in SoS? If so, what are they?

F_7: Is the primary study restricted to any type of SoS?

3.3 Conducting

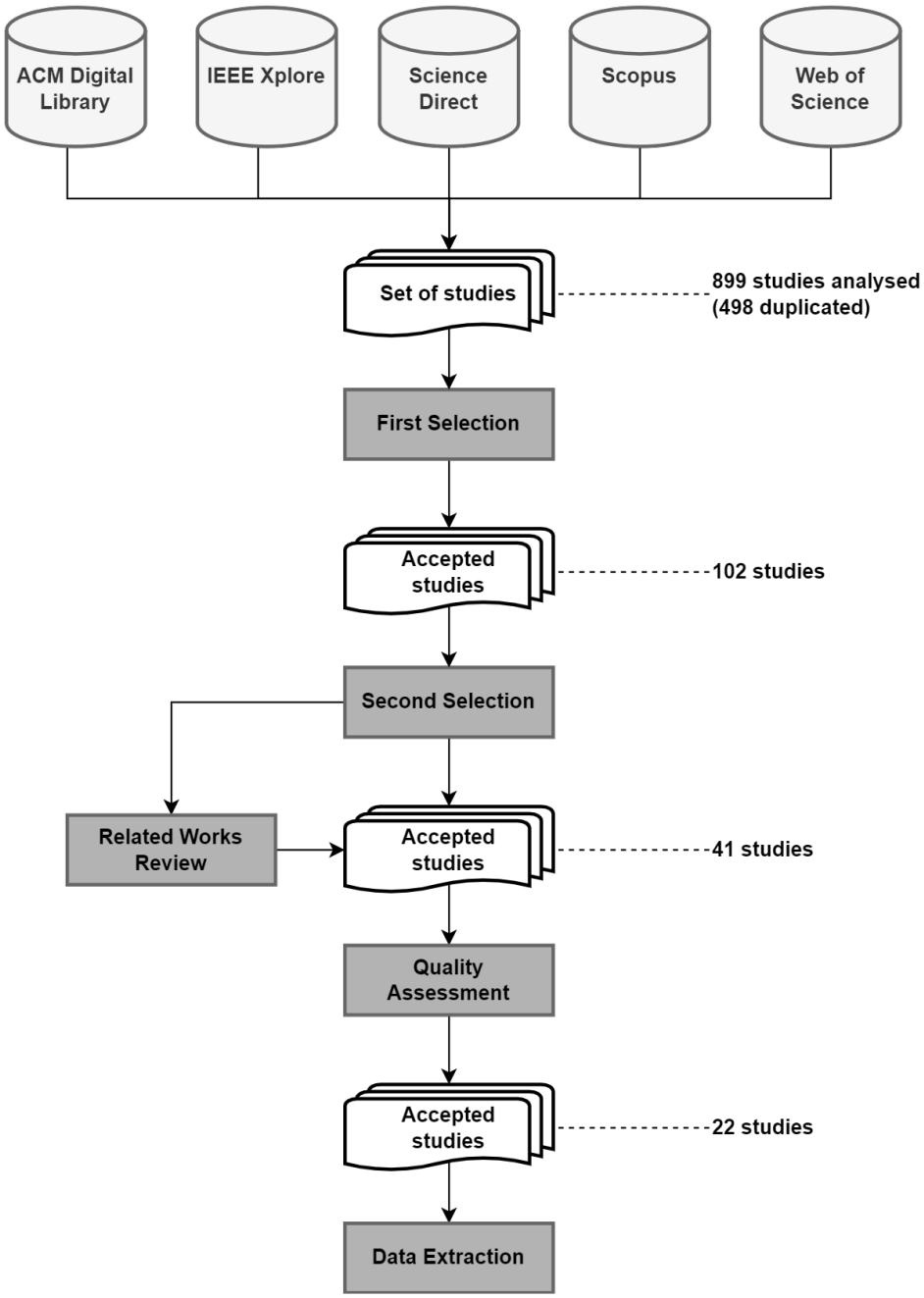
This systematic mapping was conducted from March 2019 to September 2019. It is important to observe that a formal update of this mapping would be required to include more recent works. However, such an update is left as future work due to timing restrictions. The mapping accomplished its primary goal of obtaining state of art to allow the definition of the present proposal. Nevertheless, recent works were found and studied after the publication of this systematic mapping and are referenced along the text.

In the conduction phase, primary studies were selected and evaluated by considering the following procedure for the study selection: First Selection (Section 3.3.1); Second Selection (Section 3.3.2); Quality Assessment (Section 3.3.3) and Data Extraction (Section 3.3.4).

3.3.1 First Selection

The defined search string was adapted for each search database to obtain the primary studies related to the research topic, resulting in 1387 studies. After removing duplicate studies (i.e., 498 studies), 889 were analyzed by reading their titles, abstracts, and keywords and considering the inclusion and exclusion criteria. At the end of this stage, 102 primary studies that were further analyzed in the Second Selection were selected, as illustrated in Figure 9.

Figure 9 – Conducting Phase.



Source: Elaborated by the author.

Figure 10 – Word Cloud.

Source: Elaborated by the author.

3.3.2 Second Selection

Here, the full reading of each primary study obtained in the First Selection was performed, and it was analyzed, one by one, by considering the defined inclusion and exclusion criteria. Before reading each primary study, a word cloud shown in Figure 10 was created with the keywords of these primary studies to verify whether they correspond to the terms used in the search string. From the analysis of Figure 10, it was verified that the words *risk* and *SoS* from the search string appear in the majority of the analyzed primary studies. After that, each primary study was thoroughly read. In parallel, the related works were analyzed and selected based on the inclusion and exclusion criteria. Finally, it was selected 41 primary studies for the Quality Assessment stage (see Figure 9).

3.3.3 Quality Assessment

The checklist with seven QC exposed in Section 3.2.4 was used to analyze the quality of the primary studies selected in the Second Selection. Each QC was analyzed according to the following point scale:

- The primary study does not meet these quality criteria (0 point).

- The primary study meets the quality criteria to some extent (0.5 point);
- The primary study fully meets a given quality criterion (1 point);

It was obtained the total quality score of each primary study based on the following classification: 0 - 1.0 (very poor); 1.1 - 2.0 (poor); 2.1 - 3.0 (fair); 3.1 - 4.0 (average), 4.1 - 5.0 (good), 5.1 - 6.0 (very good), and 6.1 - 7.0 (excellent). In the end, it was selected 22 selected primary studies with a score greater than or equal to 3.0 for filling out the Data Extraction Form.

3.3.4 Data Extraction

In this stage, the relevant data from each primary study accepted in the Quality Assessment was extracted by filling the Data Extract Form presented in Section 3.2.5.

3.4 Reporting

To answer the SMRQs, 22 selected primary studies exposed in Table 4 during the Data Extraction stage were fully read. The Data Extraction Forms were analyzed to answer each SMRQ qualitatively. Thus, the answers for each SMRQ are described in the following.

3.4.1 Answer to $SMRQ_1$

According to Shah, Davendralingam and DeLaurentis (2015), Baumgart, Fröberg and Punnekkat (2017), existing risk management approaches at the CS level are ineffective in the context of SoS due to the high complexity, evolutionary nature, and SoS level interactions. Concerning risk management at the CS level, Pinto, McShane and Bozkurt (2012) argue that, when designing CSs, tools and methodologies are available to solve defined problems and, as system boundaries are fixed, and expected behavior is known, the scope of these problems and their associated risks are relatively straightforward. However, the SoS boundary is not necessarily static, CSs may not have been identified, and behavior is emergent. Gandhi, Gorod and Sauser (2012) argues that in the traditional approach to risk management focused on managing risks for CSs, there is a tendency to analyze risks individually without taking into account the holistic view of interactions between the potential risks at the CS level and their consequences to the SoS. In addition, risk management approaches at the CS level do not consider the influence of external factors and constraints that may affect the overall risk associated with SoS (CONROW, 2005). An example that risk management at the CS level is incompatible with the risk management at the SoS level is the Integrated Deep Water System (IDS) presented in Gandhi, Gorod and Sauser (2012). The IDS is an SoS that applies traditional risk management practices. The IDS was broken down into “manageable parts” to facilitate the risk management of each part. However, the interdependence between IDS parts and the factors were not considered. As a result, the IDS

Table 4 – Primary studies selected in the Conducting phase.

ID	Title and Reference	Year
1	Abandonment: A natural consequence of autonomy and belonging in systems-of-systems (SALADO, 2015)	2015
2	A case for dynamic risk assessment in NEC Systems of Systems (AITKEN; ALEXANDER; KELLY, 2010)	2010
3	A conditional value-at-risk approach to risk management in system-of-systems architectures (SHAH; DAVENDRALINGAM; DELAURENTIS, 2015)	2015
4	A model based approach to system of systems risk management (KINDER; HENSHAW; SIEMIENIUCH, 2015)	2015
5	Analyzing hazards in System-of-Systems: Described in a quarry site automation context (BAUMGART; FRÖBERG; PUNNEKKAT, 2017)	2017
6	A risk and threat assessment approaches overview in autonomous Systems of Systems (ČAUŠEVIĆ, 2017)	2017
7	A risk modelling approach for a communicating system of systems (AITKEN; ALEXANDER; KELLY, 2011)	2011
8	Assessing System of Systems security risk and requirements with OASoSIS (KI-ARIES *et al.*, 2018)	2018
9	A systemic approach to managing risks of SoS (GANDHI; GOROD; SAUSER, 2012)	2012
10	Developing a methodology to support the evolution of System of Systems using risk analysis (LOCK, 2012)	2012
11	Dynamic risk management for cooperative autonomous medical cyber-physical systems (LEITE; SCHNEIDER; ADLER, 2018)	2018
12	Exile: A natural consequence of autonomy and belonging in Systems-of-Systems (SALADO, 2016)	2016
13	Identification and management of risks of system of systems (PROCHAZKOVA, 2013)	2013
14	Modeling complex systems of systems with Phantom System Models (HAIMES, 2012a)	2012
15	Responsibility modeling for identifying sociotechnical threats to the dependability of Coalitions of Systems (GREENWOOD; SOMMERVILLE, 2011a)	2011
16	Responsibility modeling for the sociotechnical risk analysis of Coalitions of Systems (GREENWOOD; SOMMERVILLE, 2011b)	2011
17	Risk management for systems of systems (CONROW, 2005)	2005
18	Risk modeling of interdependent complex Systems of Systems: Theory and practice (HAIMES, 2017)	2017
19	System of systems perspective on risk: Towards a unified concept (PINTO; MCSHANE; BOZKURT, 2012)	2012
20	Systems-based guiding principles for risk modeling, planning, assessment, management, and communication (HAIMES, 2012b)	2012
21	Systems engineering guide for Systems of Systems version 1.0 (KRISTEN, 2008)	2008
22	Towards a risk analysis method for systems-of-systems based on systems thinking (AXELSSON; KOBETSKI, 2018a)	2018

Source: Elaborated by the author.

project exceeded the budget by US$ 7 billion. Therefore, SoS-level risk management approaches are needed and essential to the success of an SoS.

One of the essential characteristics of SoS level risk management is holism because SoS takes into account the interactions and interdependence between CSs and external factors (HAIMES, 2012b). According to the DoD SoSE Guide (KRISTEN, 2008), managing CS risks does not guarantee that SoS risks are effectively managed. So, it is necessary to use both risk management at the CS level to manage risks specific to CSs and risk management at the SoS level to manage risks specific to SoS (KINDER; HENSHAW; SIEMIENIUCH, 2015). Thus, SoS risks should be managed holistically and not only considered risk management in the context of CSs. Pinto, McShane and Bozkurt (2012) mention that CS level risks tend to be managed qualitatively, while SoS level risks must be quantitatively managed due to their inherent complexity. Some primary studies present quantitative approaches in the context of SoS (KINDER; HENSHAW; SIEMIENIUCH, 2015; SHAH; DAVENDRALINGAM; DELAURENTIS, 2015; LOCK, 2012).

Another feature that SoS level risk management must take into account is the dynamics associated with the process. In project management of CSs, a CS takes months or years to complete, whereas SoS can be quickly deployed to meet an emergent requirement (KINDER; HENSHAW; SIEMIENIUCH, 2015; GANDHI; GOROD; SAUSER, 2012). Therefore, SoS risks should be managed in real-time, and SoS level risk management should be part of the SoS life-cycle decision-making process (HAIMES, 2012a). In Aitken, Alexander and Kelly (2010), they investigate the dynamic risk assessment in a Network Enabled Capability (NEC) SoS, and the authors concluded that such assessment could generate crucial information for accurate real-time decision making. However, this may also pose additional risks to operation, i.e., if the information is not presented effectively to the user, it may lead to ambiguity in the SoS.

Regarding the framework of risk management processes, there are differences between CSs and SoS. In CSs, there are risk management process frameworks defined in the literature, such as the one proposed by the PMBOK Guide (PMI, 2017), which defines a framework of seven processes (Plan Risk Management, Identify Risks, Perform Qualitative Risk Analysis, Perform Quantitative Risk Analysis, Plan Risk Responses, Implement Risk Responses, and Monitor Risks). In turn, ISO 31000 standard (ISO, 2009) also comprises seven processes (Communication and Consultation, Establishing the context, Risk identification, Risk analysis, Risk assessment, Risk treatment, and Monitoring and Review). In the context of SoS, Conrow (2005) presents an overview of risk management processes comprising: (i) Risk Planning; (ii) Risk Assessment; (iii) Risk Handling; (iv) Risk Monitoring; and (v) Risk Documentation. According to Prochazkova (2013), there are many classic methods, tools, and techniques for identifying, analyzing, and assessing risks at the CS level. However, the tools, methods, and techniques for identifying, analyzing, and managing the risks that cause or may cause different cascading failures in SoS functionality are unknown.

In order to assist with SoS risk management, some primary studies suggest a set of

questions to be answered throughout the project. In Haimes (2017), the following questions must be answered: (i) What can be done, and what options are available?; (ii) What are the compensations for all associated costs, benefits, and risks?; and (iii) What are the impacts of current decisions on future options? In Pinto, McShane and Bozkurt (2012), the following questions should guide the risk management: (i) What are the desirable events at a given time?; (ii) What can go wrong?; (iii) What are the consequences?; (iv) What is the chance of occurrence?; (v) What can be done to manage them?; and (vi) What are the alternatives?. From the differences between risk management for CSs and SoS shown in Table 5 and the comparison between PMBOK risk management processes presented in Table 6, it is concluded that although there are distinct characteristics between risk management for SoS and CSs, risk management processes for both CSs and SoS are similar.

Table 5 – Differences between risk management for CSs and SoS.

Characteristic	**CS**	**SoS**
Existing Approaches	Well established	Not well established
Scope	Static	Dynamic
Risk Analysis	Qualitative	Quantitative
Risk Monitoring	Along the project	Real time

Source: Elaborated by the author.

Table 6 – Comparison between PMBOK risk management processes (PMI, 2017) and SoS risk management processes exposed in Conrow (CONROW, 2005).

PMBOK processes	**Processes from Conrow's work equivalent to PMBOK processes**
Plan Risk Management	Risk Planning
Identify Risks	Risk Assessment
Qualitative Risk Analysis	Risk Assessment
Quantitative Risk Analysis	Risk Assessment
Plan Risk Responses	Risk Handling
Implement Risk Responses	Risk Handling
Risk Monitoring	Risk Documentation

Source: Elaborated by the author.

3.4.2 Answer to $SMRQ_2$ and its Sub SMRQs

The answer to the $SMRQ_2$ (*which are the key SoS risks?*) as well as $SMRQ_{2.1}$ (*which are the approaches used to manage these risks and their strengths and weaknesses?*) and $SMRQ_{2.2}$ (*in which domains have these risks occurred?*) was divided into two parts exposed as follows.

3.4.2.1 SoS Risks Found in the Primary Studies Analyzed

The question, "Does the primary study mention risks that may arise in SoS projects? If so, what are they?" was answered from the analysis of the data extraction form from the primary studies analyzed. Thus, the following SoS-specific risks were identified:

Abandonment: the risk of abandonment relates to the possibility of one or more CSs stop providing their services to the SoS. Abandonment is a natural and necessary consequence of the inherent characteristics of SoS autonomy, and belonging (SALADO, 2015);

Exile: the risk of exile is similar to the risk of abandonment, consisting of the probability of the SoS expelling one or more CSs. Similarly to the risk of abandonment, the risk of exile is a natural and necessary consequence of the inherent characteristics of autonomy and belonging of an SoS (SALADO, 2016);

Sociotechnical Risks: sociotechnical risks are important factors in the analysis of the risks associated with Coalition of Systems (CoS), as personal interests that hold the systems together can be fragile. For example, a CoS formed around the use of a commercial and public cloud infrastructure may be sensitive to changes in pricing or service offerings. Therefore, in addition to understanding the technical risks, it is important to understand the sociotechnical dependencies between the parts, the types of changes that may disrupt the coalition and the liabilities that may be incurred. Responsibility modeling for the Sociotechnical Risk Analysis of CoSs is a tactical level risk analysis approach that uses the concept of responsible agents (Human/organizational) and their interactions to represent a situation and identify risks in terms of a failure of agents to fulfill their responsibilities (GREENWOOD; SOMMERVILLE, 2011a);

Risks related to Dependability: according to Sommerville *et al.* (2006), dependability is a property of the system in which trust is justified by its provided services. The concept of dependability encompasses: ***availability***, i.e., readiness for correct service, ***reliability***, i.e., continuity of correct service, ***safety***, i.e., absence of catastrophic consequences on the user(s) and the environment, ***integrity***, i.e., absence of improper system alterations, ***maintainability***, i.e., the ability to undergo modifications and repairs, and ***security***, which is a composite of ***confidentiality*** (i.e., the absence of unauthorized disclosure of information), ***integrity***, and ***availability*** (AVIZIENIS *et al.*, 2004b). Addressing security demands the concurrent co-existence of *i)* availability for authorized actions only, *ii)* confidentiality, and

iii) integrity with "improper" meaning "unauthorized". Dependability threats are events or conditions that affect the CoS, such as availability, maintainability, safety, integrity, and maintainability (SOMMERVILLE *et al.*, 2006; AVIZIENIS *et al.*, 2004b). In the context of dependability, sociotechnical threats are important factors in analyzing CoS dependability because the overlapping of self-interests that hold the system together could be fragile (GREENWOOD; SOMMERVILLE, 2011a);

Evolutionary Risks: the evolution of an SoS involves continually modifying CSs in the light of business opportunities, changing circumstances, and continuous optimization. This evolution can coincide at multiple levels and areas of the SoS. The evolution of any part of an SoS may lead to undesirable emergent behavior that needs to be identified, analyzed, and mitigated (LOCK, 2012);

Hazards: hazards can lead to catastrophic consequences, such as injury, illness, or death of personnel; damage or loss to a system, equipment, or property; or damage to the environment. Hazards should be managed to ensure the safety and security of the SoS (BAUMGART; FRÖBERG; PUNNEKKAT, 2017; ČAUŠEVIĆ, 2017; AITKEN; ALEXANDER; KELLY, 2011; KI-ARIES *et al.*, 2018). HISoS is a structured process for identifying system hazards and illustrating the process in the industrial context. HISoS was applied in a quarry automation context (BAUMGART; FRÖBERG; PUNNEKKAT, 2017). Regarding other key risks mentioned, hazards are more generic because risks related to abandonment or exile can generate hazards that can lead to failures during SoS execution (BAUMGART; FRÖBERG; PUNNEKKAT, 2017; ČAUŠEVIĆ, 2017; AITKEN; ALEXANDER; KELLY, 2011; KI-ARIES *et al.*, 2018).

The risks related to evolution, abandonment, exile, sociotechnical, and dependability may raise in SoSs from automotive (AXELSSON; KOBETSKI, 2018a), aerospace (KINDER; HENSHAW; SIEMIENIUCH, 2015; AITKEN; ALEXANDER; KELLY, 2010; LOCK, 2012; HAIMES, 2017), defense (CONROW, 2005; GOROD; SAUSER; BOARDMAN, 2008; DAHMANN; JR; LANE, 2008; SIMPLEMAN *et al.*, 1998; HAIMES, 2012a; HAIMES, 2012b; AITKEN; ALEXANDER; KELLY, 2011; GANDHI; GOROD; SAUSER, 2012; HAIMES, 2017; REDMOND, 2007), mining and quarry site automation (BAUMGART; FRÖBERG; PUNNEKKAT, 2017; ČAUŠEVIĆ, 2017), medical cyber-physical systems(LEITE; SCHNEIDER; ADLER, 2018), and cloud IT infrastructure(GREENWOOD; SOMMERVILLE, 2011a; GREENWOOD; SOMMERVILLE, 2011b).

3.4.2.2 Approaches Identified in the Primary Studies to Support Risk Management

In order to manage the SoS-specific risks, these risks need to be appropriately identified and modeled to support a better understanding of their impacts on the SoS life cycle (HAIMES, 2017). Aitken, Alexander and Kelly (2011) proposed a risk modeling approach to support the

identification of threats to the safety of communication in the SoS. In this approach, risks are represented through a standard fault tree. Therefore, fault trees are used to support risk analysis. A disadvantage of this technique is that it is restricted to static analysis of SoS since such an approach does not consider the dynamic nature of an SoS. Gandhi, Gorod and Sauser (2012) have proposed an approach to support the modeling of systemic SoS risks. Systemic risk may be greater than the sum of the individual constituent risks (Schedule Risk, Technical Risks, Vendor Risk, Culture Risk, Reputation Risk, Intellectual Property Risk, Flexibility Risk, Compliance Risk, and Quality Risk). The constituent risks are thought to be interconnected, and none of them are mutually exclusive.

Concerning primary studies that encompass the risk management processes proposed in Conrow (2005), some primary studies focus on more than one risk management process, such as in Kinder, Henshaw and Siemieniuch (2015). In contrast, other primary studies focus on specific processes, such as risk identification processes (PROCHAZKOVA, 2013; GREENWOOD; SOMMERVILLE, 2011a), risk analysis (BAUMGART; FRÖBERG; PUNNEKKAT, 2017; AXELSSON; KOBETSKI, 2018b) and risk assessment (KI-ARIES *et al.*, 2018; LEITE; SCHNEIDER; ADLER, 2018). The key features of such approaches and the risks that such approaches are intended to manage are outlined in the following.

Responsibility modeling for identifying socio-technical threats to the dependability of COSs: The responsibility modeling approach aims to identify socio-technical threats, which is seen as complementary to the earlier approaches discussed and more general approaches to risk analysis. Responsibility modeling uses the concept of responsible agents (Human/Organizational agents) and their interactions to represent a situation and identify risks (GREENWOOD; SOMMERVILLE, 2011a);

Responsibility Modeling for the Sociotechnical Risk Analysis of CoSs: It is a tactical level risk analysis method to use the concept of responsible agents and their interactions to represent a situation and identify risks in terms of agents failing to fulfill responsibilities. Currently, the authors seek to extend the approach to the scope of analysis to include stakeholder views and values that the migration to a systems coalition will impact. Thus, including an additional dimension to risk analysis beyond risk analysis from a mechanistic means perspective, where human agents are assumed to be passive and compatible. Instead, it also takes into account factors that make stakeholders resist and conflict with change, which represents an important class of risk that is missing from this analysis (GREENWOOD; SOMMERVILLE, 2011b);

OASoSIS: This approach aims to align appropriate SoS factors and concepts to obtain, analyze, and validate security risks using tool support in the context of SoS. OASoSIS aims to provide a simple, repeatable, and reusable process for identifying security risks in an SoS. However, due to independent SoS collaborations, there will always be an element

of information based on unknown or unavailable risks to be assessed. Interoperability between dependent systems is difficult to achieve on an SoS without understanding the SoS. Therefore, it is important to understand the minimum level of information required to make a satisfactory assessment of security risks, certainly when converting them to security requirements (KI-ARIES *et al.*, 2018);

Hazards In Systems-of-Systems (HISoS): HISoS is a structured process for identifying SoS hazards and illustrating the process in the context of the industrial example provided (quarry automation context). The authors have used the Redmond classification (REDMOND, 2007) to categorize the identified hazards. However, such classification is insufficient to deal with the example used in their work (BAUMGART; FRÖBERG; PUNNEKKAT, 2017);

A methodology to support the evolution of SoS using risk analysis: it is a decentralized SoS methodology that explores the types of information that need to be collected and discussed at the meta-level to analyze SoS risks, along with the development process, to assist end users in managing generic evolution. SoS evolution life-cycle is designed to describe how this information can be used. The present methodology uses a modified form of the HAZOP to analyze the risks associated with evolution (LOCK, 2012). The author suggests that appropriate tools should be developed to collect, model, manage and disseminate SoS information at the meta-level, as this is important for integrating different modeling and risk analysis tools into the overall methodology for analyzing SoS evolution;

Risk Management Approach for Autonomous and Cooperative Medical CPSs: It supports the management of run-time risks for standalone and cooperative Cooperative Medical CPSs. The approach consists of a holistic risk assessment model that considers relevant safety parameters with a syntax based on Bayesian probability networks. This risk model supports a complete risk assessment and classification of hazardous situations, enhancing standalone decisions and the reconfiguration process in run-time security certification. Therefore, the presented solution considers the guarantees of a configuration in the risk assessment, thus improving the detection of risk situations. In contrast, the system availability is improved due to the identification of adequate system guarantees appropriate to the risk of the situation. According to the authors, it is still needed to refine the risk model and expand it for use in other types of hazards (LEITE; SCHNEIDER; ADLER, 2018);

Model based approach to SoS risk management: A model-based approach built upon a central Bayesian Belief Network (BBN) to represent risks and contributing factors. In order to allow probabilities within BBN to be filled objectively rather than subjectively, the authors propose the usage of supporting models, which are executed using a Monte Carlo model-simulation approach, thus generating results that can be 'learned' by the BBN. The present approach is still in the conceptual modeling phase, and it needs to be validated in

other contexts to verify its generic applicability (KINDER; HENSHAW; SIEMIENIUCH, 2015);

SoS CVaR Optimization: It is a conditional value-at-risk approach for managing a systems portfolio in an SoS context. The framework can potentially be adapted to a tool that allows decision-makers at various levels of the SoS hierarchy to mitigate extreme risks that result from complicated interdependence between CSs. The approach considers the complexity of potentially nonlinear behaviors through an approximation-based structure that results in a linear system. Another attractive feature of the author's approach is that the formulation can be adapted to use direct simulation or observed data. The application of such portfolio-based approaches to the SoS domain is limited, but this primary study illustrates the versatility of the method (SHAH; DAVENDRALINGAM; DELAURENTIS, 2015);

Identification and Management of Risks of SoS: It is a method for identifying, analyzing, assessing, and managing cross-cutting risks based on experience and principles of good engineering practice. According to the author, although the approach does not extensively use highly sophisticated application methods, it can be time-consuming due to the high-quality data requirements, which are always challenging to collect and experience. The author's approach is pragmatic and transparent, and it provides good results that practice the object management needs of SoS (PROCHAZKOVA, 2013);

Towards a risk analysis method for SoSs based on systems thinking: This method extends the existing Systems-Theoretic Accident Model and Processes (STAMP) (LEVESON, 2004) safety analysis method based on systems thinking. STAMP is intended to deal with risks other than safety, and it was adapted to SoS. The method aims to derive requirements from CSs to reduce emerging risks in the SoS. According to the authors, the present method is still under development, so it needs to be applied in other contexts. In addition, the method does not support quantitative analysis and does not deal with the dynamic evolution of an SoS, besides lacking automated tooling support (AXELSSON; KOBETSKI, 2018b).

The majority of the identified SoS risk management approaches focuses in the defense domain (CONROW, 2005; GOROD; SAUSER; BOARDMAN, 2008; DAHMANN; JR; LANE, 2008; SIMPLEMAN *et al.*, 1998; HAIMES, 2012a; HAIMES, 2012b; AITKEN; ALEXANDER; KELLY, 2011; GANDHI; GOROD; SAUSER, 2012; HAIMES, 2017; REDMOND, 2007). It was also identified SoS risk management approaches validated in the automotive (AXELSSON; KOBETSKI, 2018a), aerospace (KINDER; HENSHAW; SIEMIENIUCH, 2015; AITKEN; ALEXANDER; KELLY, 2010; LOCK, 2012; HAIMES, 2017), mining and quarry site autonomous systems(BAUMGART; FRÖBERG; PUNNEKKAT, 2017; ČAUŠEVIĆ, 2017), cooperative medical cyber-physical systems(LEITE; SCHNEIDER; ADLER, 2018), and cloud

IT infrastructure(GREENWOOD; SOMMERVILLE, 2011a; GREENWOOD; SOMMERVILLE, 2011b) domains.

3.5 Threats to Validity

The main threats identified to the validity of this systematic mapping were:

Study Selection: From the six databases selected based on recommendations of Dyba, Dingsoyr and Hanssen (2007), Kitchenham and Charters (2007), Springer was not used because it does not have the function of searching studies only by metadata. In order to improve the quality of the study selection, the Related Works Review was performed. However, it could be possible that important primary studies can be missed;

Data extraction: The extracted data from the primary studies using the Data Extraction Form, and the reviewer's personal opinion might influence such extraction. Thus, any doubts regarding data extraction were discussed with an SoS expert. However, this discussion may not have been sufficient;

Quality Assessment: The primary studies were assessed through seven QCs. However, the reviewers' opinions may have influenced the assignment of scores.

3.6 Final Considerations

During the project or operation of any SoS, risks may arise and lead to catastrophic consequences causing failures in the project. Therefore, it is essential to manage such risks correctly. In this context, a systematic mapping was performed to identify the principal risks related to SoS and the main approaches used to manage SoS risks. By providing answers to the SMRQs, it is clear that there are different types of SoS risks, such as abandonment or risks related to dependability. However, the existing approaches to managing SoS-specific risks are not well established because they were not applied in several domains and are still under development. Even though risks can lead to catastrophic consequences to the SoS, no approaches and tools are available to support identifying, analyzing, and managing such risks. The same scenario was described in Chapter 2, since existing compositional techniques and tools that support safety analysis for complex systems do not consider SoS characteristics. Therefore, existent compositional safety analysis approaches can be extended to the SoS-level, considering that SoS failures can be composed of the characterizations of individual CSs. Chapter 4 and Chapter 5 seek to cover the gap identified by proposing a novel approach to performing SoS safety analysis.

CHAPTER

4

SOSSAFE META-MODEL

4.1 Initial Considerations

To answer RQ_2, this chapter presents the SoSSafe meta-model, which was proposed in this work to support SoS architectural design and safety analysis. The SoSSafe meta-model was built upon the analysis of: i) existing *Systems of Systems* safety analysis and risk assessment processes, e.g. Redmond SoS safety analysis methodology (REDMOND, 2007; REDMOND; MICHAEL; SHEBALIN, 2008); ii) the systematic mapping study presented in Chapter 3 (LOPES *et al.*, 2020); iii) state-of-the-art Model-Based Design techniques, such as UML version 2.5 (OMG, 2017b), System Modeling Language (SysML) version 1.6 (OMG, 2017a) Object Management Group specifications, Papyrus UML (Eclipse Foundation, 2017), MATLAB Simulink (MathWorks, 2021), and AADL (SAE, 2017); and iv) safety models provided by state-of-the-art model-based safety analysis frameworks, such as HiP-HOPS (PAPADOPOULOS *et al.*, 2011), AADL Error Annex, CHESS Failure Logic Analysis (GALLINA; SEFER; REFSDAL, 2014), CHESS State-Based Analysis (MONTECCHI; GALLINA, 2017), and the ODE meta-model (DEIS, 2020).

The proposed meta-model extends the ODE meta-model with modeling concepts to support SoS architectural design and safety analysis. These models can be specified with the support of state-of-the-art model-based software engineering and systems safety analysis (PAPADOPOULOS *et al.*, 2011; CMU, 2020; MAZZINI *et al.*, 2016a). In the same way, the proposed meta-model is an exchange format for SoS design and dependability models produced by different tools to support different engineering stories. The proposed meta-model supports developers in capturing various aspects of SoSs, including architecture, HARA, failure logic, FTA, and FMEA.

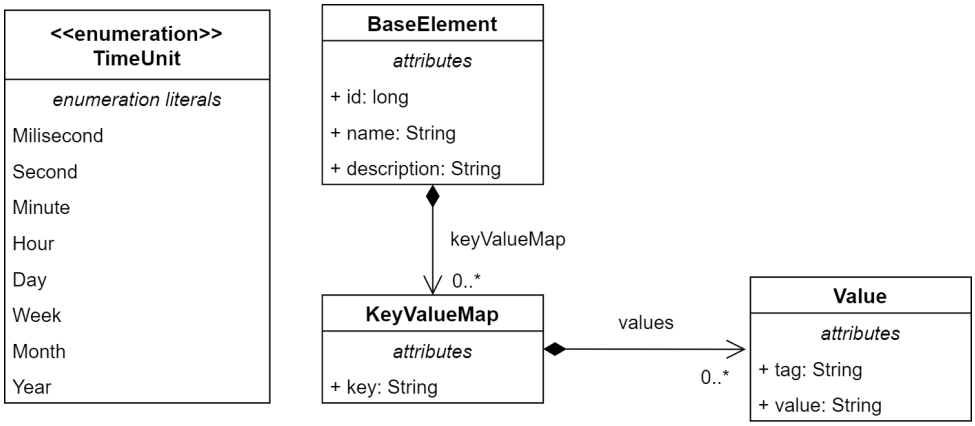
The SoSSafe meta-model packages are described in this chapter. Section 4.2 presents the *Base* package. Section 4.3 describes the *Design* package. The *HARA* package is presented in Sec-

tion 4.4. Section 4.5 presents the *Dependability* package. Section 4.6 describes the *FailureLogic* package. Section 4.7 presents the *FTA*, *FMEA*, and Markov failure logic sub-packages. Finally, Section 4.8 presents the final considerations.

4.2 Base Package

The Base package (Figure 11), built upon ODE::Base package, encapsulates *BaseElement*, *KeyValueMap*, and *Value* classes, as well as their relationships, representing *Element*, *NamedElement*, and *Property* Meta-Object Facility (MOF) (OMG, 2019) base classes and their implementations in the Eclipse Modeling Framework (EMF) (Eclipse Foundation, 2022). The *BaseElement* is the common base class of all ODE classes and *Element* is the common base class of all MOF metaclasses. A *BaseElement* may own properties, operations, and other elements. A *BaseElement* has a unique *id*, *name*, *description*, and a set of *KeyValueMap* elements. In the same way in ODE meta-model, each *KeyValueMap* in SoSSafe meta-model is used to associate a particular element with user-defined properties (*key*), whose values are retrievable through *Value* elements. For properties composed by sets of *Value* elements, *tag* can be used to further specify which *Value* element is required.

Figure 11 – Base Package.



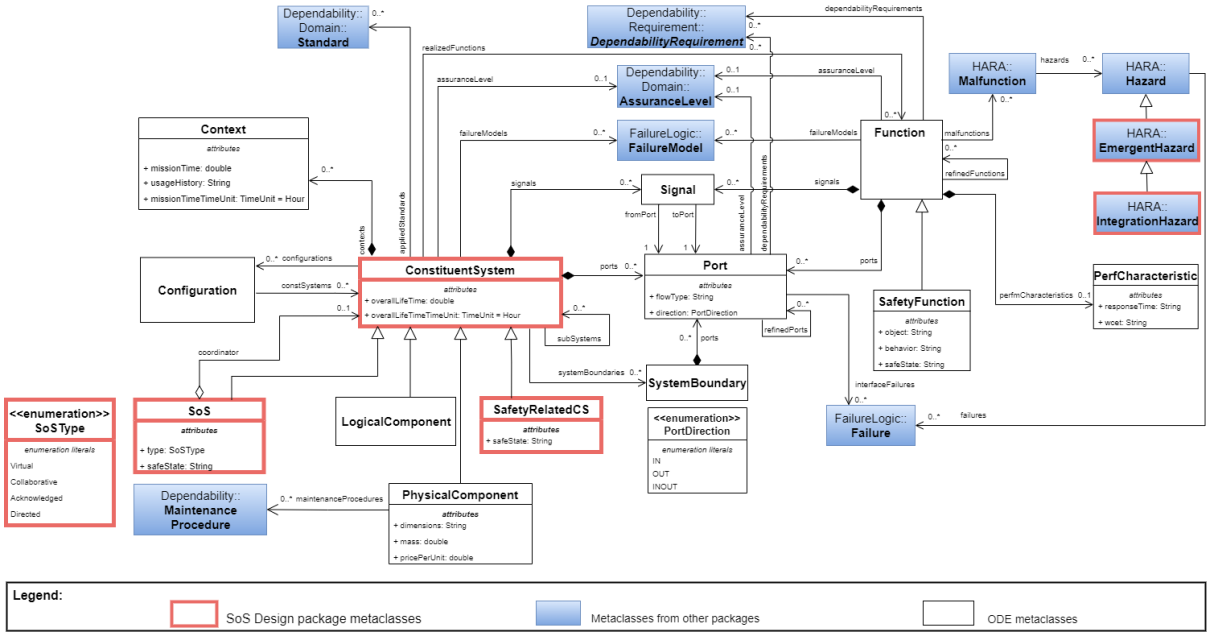
Source: Elaborated by the author.

The *KeyValueMap* allows extending the meta-model with system information. If a system or component provider or user agrees on a specific protocol to describe system or component information not formally defined yet in the meta-model, they can enrich the meta-model with such information. The Base package also includes the *TimeUnit* enumeration commonly used in other packages. *TimeUnit* enumeration defines the timing properties of an SoS, CS, or component to consider their worst-case execution time. The elements from other SoSSafe meta-model packages inherit *BaseElement* properties.

4.3 SoS Design Package

Dependability can only be demonstrated in the context of structural and behavioral models of a particular system or SoS. The concepts needed to model structural and behavioral aspects of SoSs and CSs architectures are encapsulated into the SoS Design package. This package extends the ODE::Design package with *SoS*, *ConstituentSystem*, and *SafetyRelatedCS* elements to represent SoS, its non-safety and safety-related CSs, and their *relationships* as illustrated in Figure 12.

Figure 12 – SoS Package.



Source: Elaborated by the author.

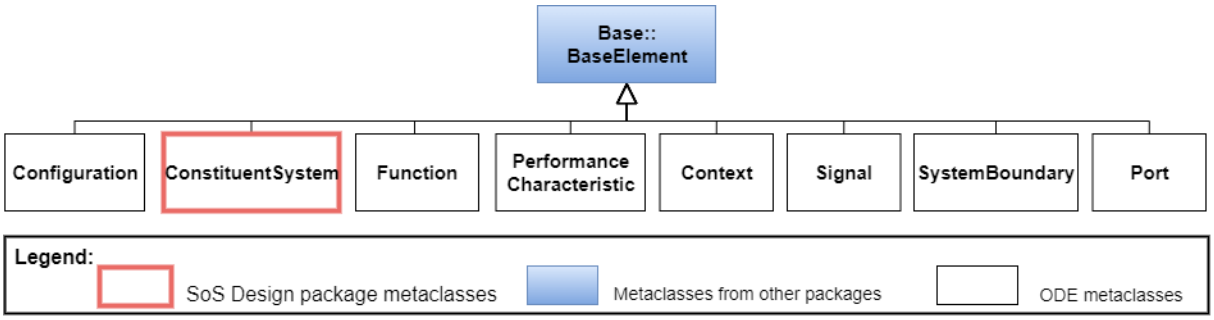
A *System* or *ConstituentSystem* can be a logical (software) or physical (hardware) representation of the system, and a *Function* is used for representing a system behavior. The *ConstituentSystem* is the root element from SoS design package, in the same way *System* is the root element from **ODE::Design** package. A *SoS-Safe::ConstituentSystem* is a composite element that may comprise other subsystems, logical and physical components, and their ports and connections (represented by *Signal* elements). Ports are explicitly defined interfaces from which the *ConstituentSystem* communicates with external systems, sub-systems, and components via signals.

A *Signal* represents a connection between ports through which the information flows (data, energy, or material) from a source to a destination system. A *Port* has an assigned direction (*PortDirection* enumeration) according to the direction of the signal, which can be incoming (IN), outgoing (OUT), or both (INOUT). The boundaries of an SoS, CS, or component is determined by their ports, which means they interface with other SoSs, CSs, components, or the *environment* and their boundaries are represented by the *SystemBoundary* metaclass.

Different phases of a system engineering lifecycle focus on different aspects of the system under design or the coalition of independent systems in an SoS, such as *logical*, *physical*, and *safety*, which demand the analysis of different attributes. We used the *LogicalComponent*, *PhysicalComponent*, and *SafetyRelatedSystem* ODE design package elements to support the specification and analysis of different SoS and CS attributes. Independent from the modeling aspect, a CS is a hierarchical representation of the architecture and has a set of *Function* elements representing the behavior a *ConstituentSystem* should realize, or the emergent behavior provided by a coalition of independent CSs in an SOS. A set of performance characteristics (*PerfCharacteristic*), i.e., non-functional requirements, can emerge from CS or SoS functional requirements and be attached to a function or emergent function (in the case of an SoS) to be realized. A *PhysicalComponent* can have several maintenance procedures (*MaintenanceProcedure*), e.g., a sensor can be repaired or replaced by another.

A *ConstituentSystem* can be a *SafetyRelatedCS* if the occurrence of a failure may lead to catastrophic damage to people, environment, or property. A *SoS* element extends *ConstituentSystem* to represent the coalition of a set of independent systems in a SoS. The SoS types are stored into the *SoSType* enumeration Figure 12. *SoS*, *ConstituentSystem*, and all design package elements inherit all basic attributes and relationships from *BaseElement* class from the *ODE::Base* package as illustrated in Figure 13.

Figure 13 – SoS Base Element and sub-types.



Source: Elaborated by the author.

An SoS, CS, or component can operate within one or more contexts. Each *Context* contains relevant information about the SoS or CS component operation, usage, or environment. An SoS or CS can be present in zero or multiple configurations. A *Configuration* may contain one or more *SoS* or *ConstituentSystem* elements. Due to the open and adaptive nature of SoSs, different configurations, which cannot be predicted at design time, can dynamically emerge and dissolve in response to environmental changes.

SoSs, CSs, components, and functions can fail and their failure behavior are captured in *FailureModel* from *FailureLogic* package. A *Function* can have malfunctions that describe safety-critical deviations from the intended behavior. In the same way, an SoS or CS may have different failure models (*FailureLogic::FailureModel*), their ports, and may have related interface

failure modes (FailureLogic::Failure) describing the failure propagation interface of a given CS that implements a *Function*. Associations between *ConstituentSystem*, *Function* elements, and *FailureLogic::FailureModels* and associations between *ODE::FailureLogic::Failure* and *Port* elements make more explicit the relationship between SoS and CS failure analysis, and individual failure behaviors at function, system, and component levels. SoS and CSs may raise different failures that can lead to the occurrence of hazards (*HARA::Hazard*) with different consequences to people, environment, or property in different configurations. Each SoS or CS configuration may have different failure models and failure propagation interfaces, i.e., failure modes.

In a SoS, accidents do not occur due to the propagation of a *failure* to an actuator output, but due to *failures* in interactions between different *constituent systems* performing a *collaborative function* (emergent behavior) together that cannot be provided by any system in isolation. For example, a Traffic Light Assistant SoS is a location and map-based service that provides traffic light information to the user based on the vehicle position and driving direction to help the user to reduce both time and CO2 emissions to reach a destination. A failure in this SoS may lead to an increase in the time to reach a destination and an increase in CO2 emissions.

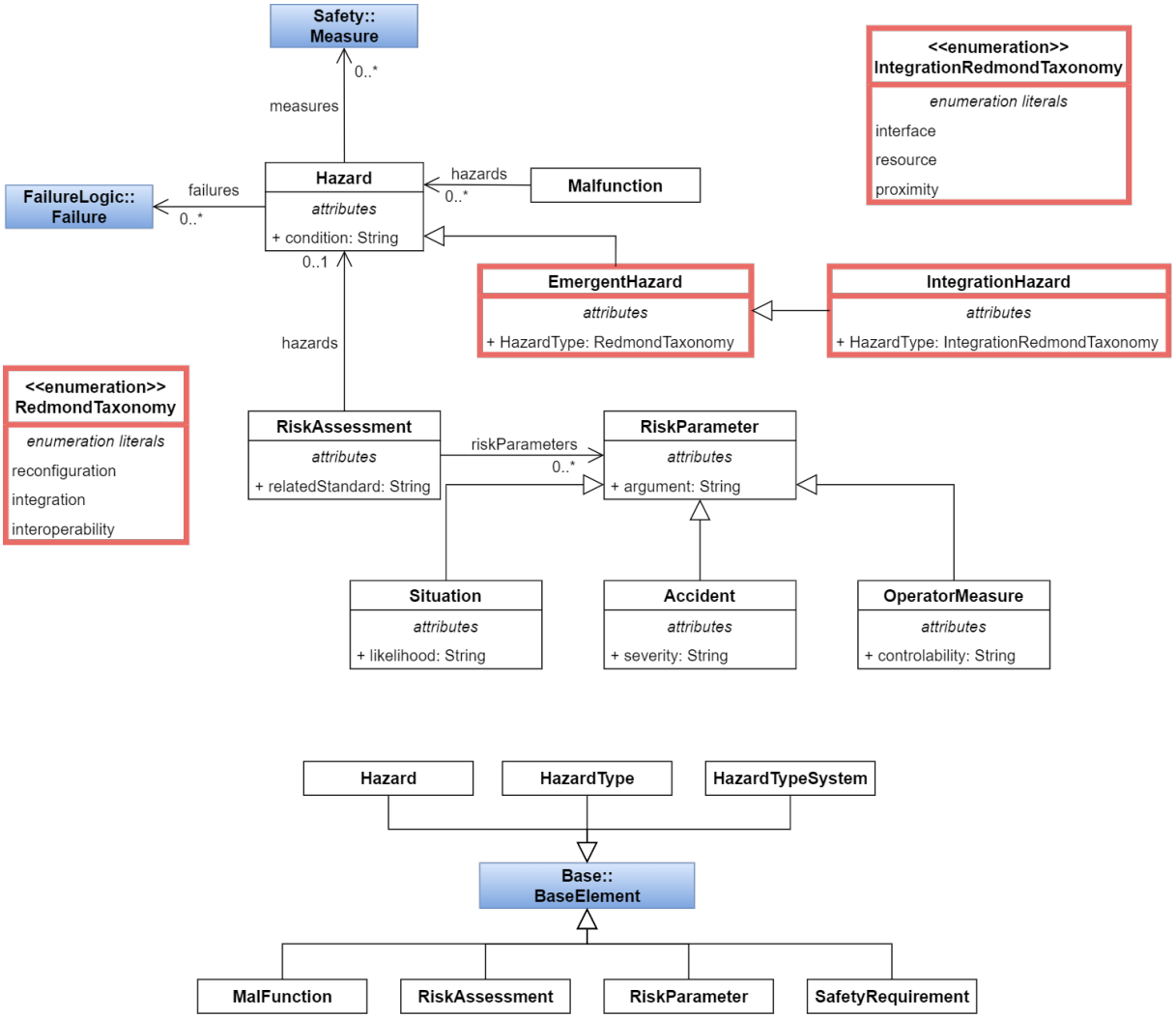
CSs should be developed in compliance with safety and security standards **Dependability::Domain::Standards**. A dependability *Standard* provides a safety/security *lifecycle* to develop systems in a given application domain, a risk-based approach for determining risk classes for classifying the risk posed by each *hazard* during risk assessment, and requirements for validation and confirmation measures to ensure that an acceptable level of safety/security is achieved. A CS or SoS can have an assurance level, which assigns a set of *Dependability::Requirements::DependabilityRequirement* elements to be addressed to achieve the given assurance Level. Standards define dependability requirements of different stringencies to be addressed to reduce the effects of hazards or interface failure modes according to the criticality of the targeted assurance level.

4.4 HARA Package

The *HARA* package extends the ODE::HARA package (DEIS, 2020) with *EmergentHazard* and *IntegrationHazard* modeling concepts and *RedmondTaxonomy* and *IntegrationRedmondTaxonomy* hazard enumeration types to support HARA at SoS CS levels as illustrated in Figure 14. HARA is the starting point for SoS and system dependability analysis where malfunctions (*Malfunction*) on the intended SoS or CS behaviors specified as *Function* elements are identified, resulting in hazards (*Hazard*). Hazard risks are estimated via levels of *likelihood* (*Situation* metaclass), *severity* (*Accident* metaclass), and *controllability* (*OperatorMeasure* metaclass), or other risk parameters associated with the risk assessment process (*RiskAssessment*) of the targeting safety standard (Dependability::Domain::Standard). *RiskParameter* elements and

their values and assurance levels may vary from one standard to another.

Figure 14 – HARA Package.



Source: Elaborated by the author.

Exposure (*likelihood*), *severity*, and *controllability* risk parameters, and their possible values defined in ISO 26262 automotive safety standard were considered in this meta-model. For the *likelihood* parameter from *Situation* metaclass, ISO 26262 defines *E1 – Very Low*, *E2 – Low*, *E3 – Medium*, *E4 – High* for classifying the probability of occurrence of a hazard or a component failure mode. For the *severity* attribute from *Accident* metaclass, ISO 26262 defines: ***S1*** *- Light and Moderate Injuries*, ***S2*** *- Severe and Life-Threatening Injuries* - Survival Probable, and ***S3*** *- Life-Threatening Injuries - Fatal Injuries* severity levels to determine the impact of the occurrence of a hazard or a component failure in terms of the damage to people, property, or environment. Finally, for the *controllability* attribute from *OperatorMeasure* metaclass, ISO 26262 defines three possible levels concerning the extent in which the operator or any other occupants can take actions to minimize *hazard* or *component failure* effects: ***C1*** *- Normal*, ***C2*** *- Simple*, and ***C3*** *- Difficult - Uncontrollable*. The risk posed by a *Hazard* or a component failure

(*FailureLogic::Failure*) can be classified into: **ASIL QM** (Quality Management), **ASIL A**, **ASIL B**, **ASIL C**, or **ASIL D** based on levels of *exposure*, *severity*, and *controllability*. Mitigating a hazard or failure with higher severity, exposure, and controllability demands a higher ASIL, which implies higher development and validation (*Dependability::Measures*) efforts and costs.

The *Hazard* metaclass represents a combination of failure conditions (FailureLogic::Failure elements) at the *Systems of Systems* or *Constituent System* level with potential to cause harm (damage or loss) to people, property, or environment. SoS-level hazards are represented by *EmergentHazard* and *IntegrationHazard* metaclasses. An emergent hazard can be classified as *Reconfiguration*, *Integration*, or *Interoperability* according its source in conformance with Redmond Taxonomy (REDMOND, 2007) (*RedmondTaxonomy* enumeration type). An *integration hazard* can be classified into: *Interface*, *Resource*, or *Proximity* hazard according to its source (*IntegrationRedmondTaxonomy* enumeration).

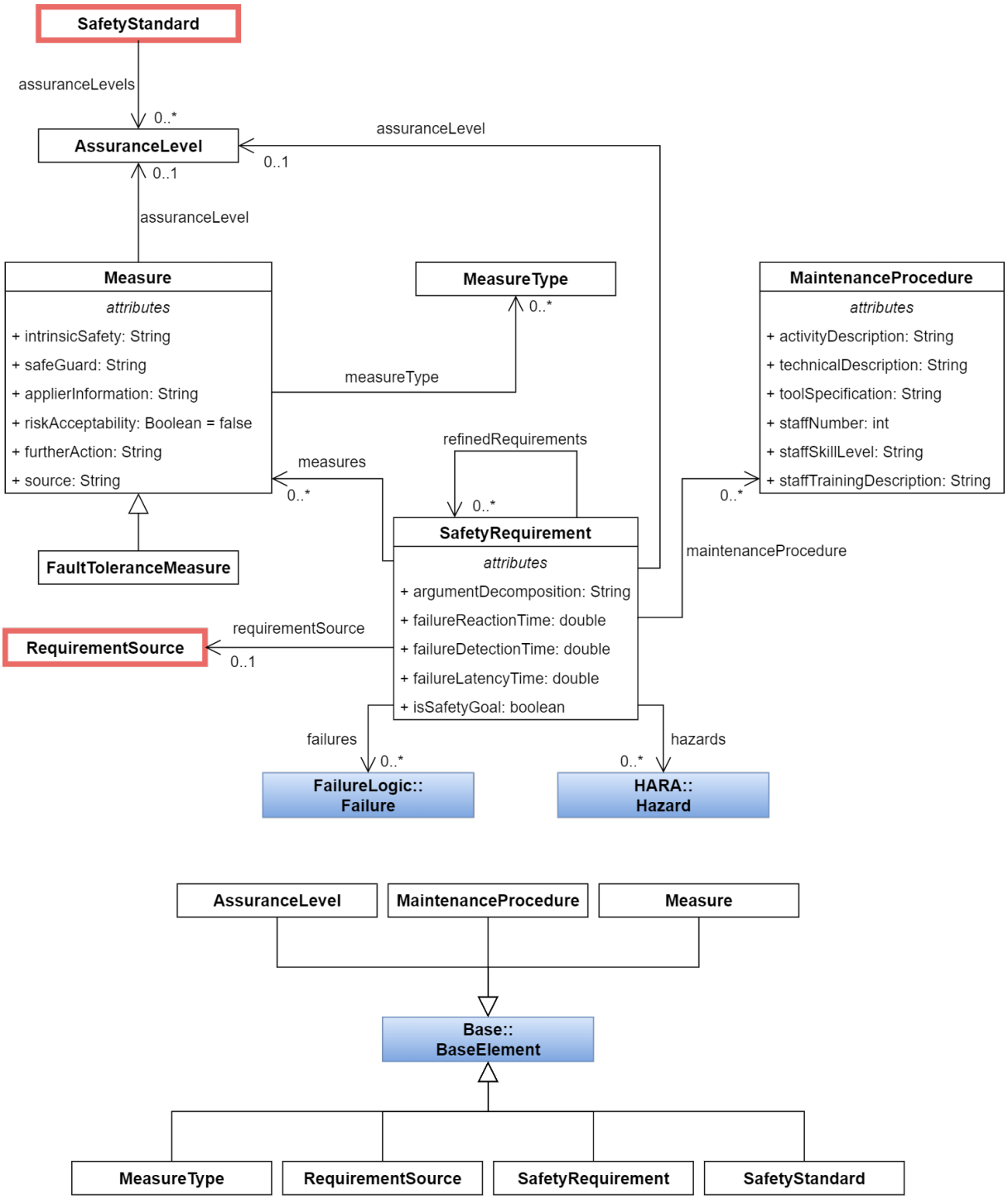
Different *Dependability::Measures* can be assigned to eliminate or minimize the effects of an SoS or a CS *Hazard* according to its risk. A *Hazard* is referenced by a *RiskAssessment* element for conducting a probabilistic risk assessment for classifying the risk posed by that hazard to the overall safety. A *Hazard* is also referenced by a *Function* due to the data exchange between them. *Hazard* is also referenced by a *Design::SafetyFunction* for the derivation of a *safety function* (see Figure 12), and by a *Dependability::Requirements::SafetyRequirement* for deriving the *safety goals* and *safety requirements*. In IEC 61508 (IEC, 2010), a safety requirement is a requirement for a *Safety Function* and its associated safety integrity (*assurance*) level. A safety function is a function to be implemented by a safety-related system intended to achieve or maintain the safe state for an electronic unit control for a specific hazardous event. A safety-related system aims to implement safety functions to ensure the safe state of an electronic unit.

4.5 Dependability::Requirements Package

The **Dependability::Requirements** package (Figure 15), reused from ODE meta-model (DEIS, 2020), contains the elements to model dependability requirements and their relationships with their requirement sources (*Dependability::RequirementSource*), e.g., product-specific goals and requirements, product and process requirements from relevant safety and security standards, and domain-specific regulations to achieve a given assurance level. A dependability requirement is a requirement for a CS or SoS to deal with the avoidance of service failures more frequently and severely than acceptable. SoS and CS failure elements and their associated attributes are handled by **Dependability::HARA** and **FailureLogic** packages.

A dependability requirement can be assigned to reduce the effects of one or more failures (*FailureLogic::Failure*). An assigned dependability requirement may require applicable measures (*Dependability::Measure* or maintenance procedures (*Dependability::MaintenanceProcedure*)

Figure 15 – Dependability Package.



Source: Elaborated by the author.

to address a specific assurance level (*Dependability::AssuranceLevel*). A safety requirement is a sub-type of dependability requirement derived from hazards (*HARA::Hazard*) to mitigate their effects on the overall system safety.

A *Measure* is an appropriate action or mechanism, e.g., fault avoidance and removal, to reduce safety and security risks to system dependability to an acceptable level. A fault tolerance measure is a risk reduction measure identified during functional hazard analysis and used to derive safety goals and requirements. A maintenance procedure is an applicable measure to reduce the risks of failures in physical (hardware) components. Fault prevention, fault detection and correction, fault tolerance, and fault forecasting are dependability measures to address safety requirements associated with assurance levels with different stringencies. Fault tolerance measures are required to reduce hazard and component failure effects, identified during HARA and fault tree analysis, with higher assurance levels, e.g., *ASIL C* and *ASIL D*. Performing the walk-through of the design is an example of a fault prevention measure recommended by ISO 26262 Part 6-7.4.8.1: System design and verification methods (ISO, 2011) for verifying the system design to achieve ASIL A. On the other hand, performing control flow and data flow analysis is fault detection and fault removal mechanisms required by ISO 26262 for verifying the design of system functions classified as *ASIL C* or *ASIL D* assurance levels.

Failure elements and associated attributes are handled by *HARA* and *FailureLogic* packages. The *DependabilityRequirement* element references a *FailureLogic::Failure* and a *Measure* in the HARA package. A dependability requirement can also be modeled as a safety (*Dependability::SafetyRequirement* element) requirement. In SoSs, we can have emergent SoS dependability requirements assigned to emergent functions, CS requirements, functional (subsystem), and technical safety requirements.

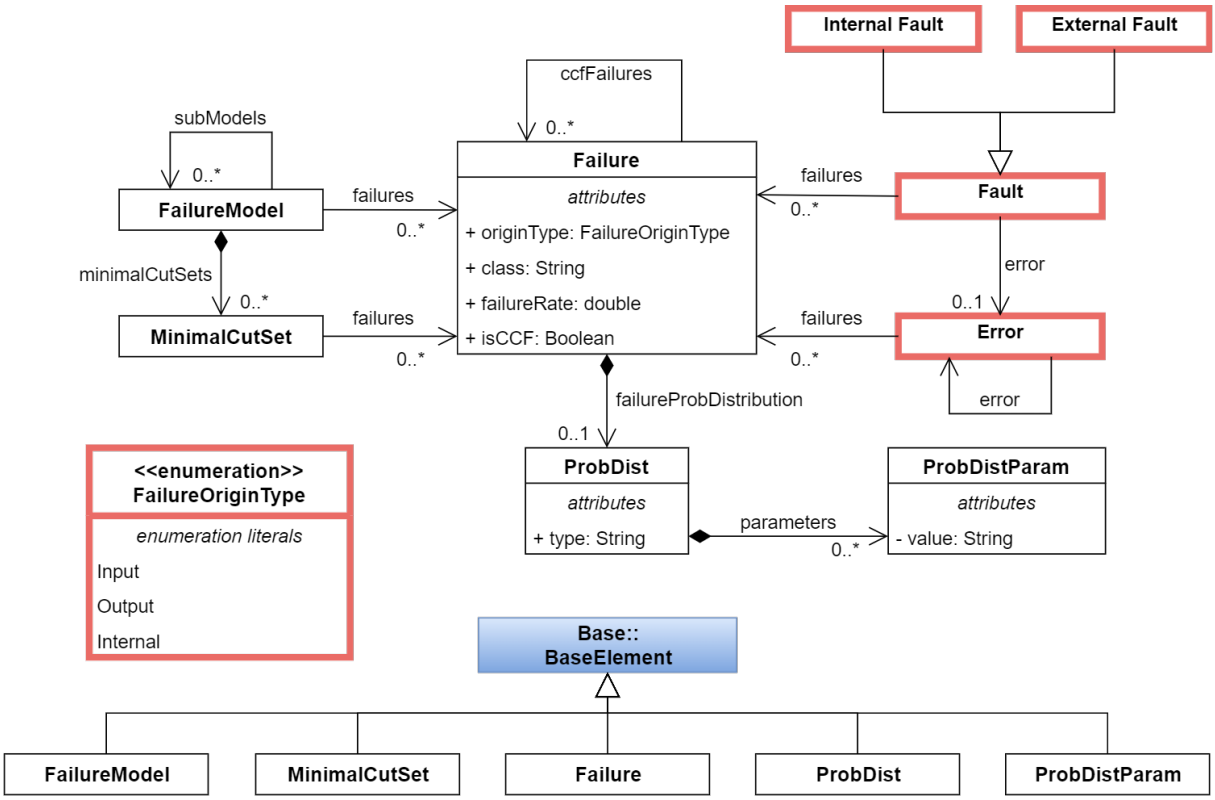
4.6 FailureLogic Package

SoS and CS safety analysis is performed based on the top-level dependability requirements derived from hazards identified during HARA, describing the potential causes of failures in an SoS, CS, or function leading to hazards to be mitigated. The ODE **FailureLogic** package and its sub-packages contain the elements to model the potential system and systems of systems failures and their causes using existing safety analysis techniques such as FTA, FMEA, and probabilistic Markov modeling (RABINER; JUANG, 1986).

The **ODE::FailureLogic** package (Figure 16) and its sub-packages provide the elements to support modular and hierarchical failure analysis based on the system design using analysis techniques such as CHESS Failure Logic Analysis (GALLINA *et al.*, 2012), HiP-HOPS (PAPADOPOULOS *et al.*, 2011), and AADL Error Annex (DELANGE, 2016). The *Failure* element abstracts common characteristics of failures within functions, systems, or components. The *ODE::FailureLogic* package (Figure 16) was extended with *Fault*, *InternalFault*, *ExternalFault*,

and *Error* metaclasses to support specification of failure data at SoS and CS levels using Failure Logic Modeling techniques during SoS safety analysis.

Figure 16 – FailureLogic Package.



Source: Elaborated by the author.

Failures in architectural elements, e.g., SoS, CS, and components, can be categorized into *input*, *output*, or *internal* according to its origin as illustrated in Figure 16. The *FailureLogic::Failure* element is useful for composing failure analysis results of hierarchical models. An SoS, CS, or component's failure behaviors are encapsulated into one or more *FailureModel* elements. A *Failure* can be a Common Cause Failure (CCF), when its occurrence can trigger other failures. A probabilistic value for a failure rate can be assigned to hardware failures. A probability distribution (*ProbDist*) model can be used to calculate the unavailability of a hardware element for a given failure. There are some probability distribution failure models, e.g., Constant Failure and Repair Rate, Mean Time to Failure and Repair, with different formulas and parameters to calculate the unavailability of a component for a given basic event. *Basic event* is a failure that is not further propagated, being the root cause of a top-event. A top event can be a failure or combination of failures leading to the occurrence of a hazard with the potential of causing harm to people, property, or the environment.

A *Failure* is characterized by its origin type (*input*, *output*, or *internal*), class, which describes its nature, e.g., data omission or commission, incorrect value, data sent too early or too late, and common cause failure (*isCCF*) attributes. A hardware *Failure* has a probabilistic

failure rate specified using a probability distribution. A *Fault* is a condition leading a logical component to fail during its execution. An *InternalFault* condition is a deviation in the component specification accidentally introduced in the source code by an incorrect human action. The introduction of an incorrect logic operator in the source code by a human mistake is an example of *InternalFault*. A *Fault* introduced in a logical component does not necessary leads to an *Error*. An *Error* is any incorrect intermediate state of a component that violates its specification. An error is an undesirable difference between desired and obtained output. An *Error* does not necessarily leads to the occurrence of a *Failure*. There exist two possible error types: computing and domain errors. A computing error is a difference between desired and obtained output. A domain error is a difference between a program's executed and expected execution path. An *ExternalFault* is an accidental or deliberated malicious action (i.e., a cyberattack) in SoS, or CS (*asset*) performed by an external agent with potential to cause harm or loss. A *Fault* can be the root cause of failures. The occurrence of an *Error*, by executing a faulty code accidentally introduced by a human mistake, may lead to another *Error* or directly cause a *Failure*.

An assurance level can be assigned to classify the risk posed by a software failure to system dependability. A component *FailureModel* may have a set of minimal cut-sets. A minimal cut-set represents the smallest possible combinations of *Failures* leading to the occurrence of a top-event, i.e., an output failure mode. Minimal cut-sets are used as the basis for failure propagation and probabilistic analyses. An input or output failure is associated with a system or component port. A hazard can be associated with several output failures combined using logical operators (AND, OR, NOT).

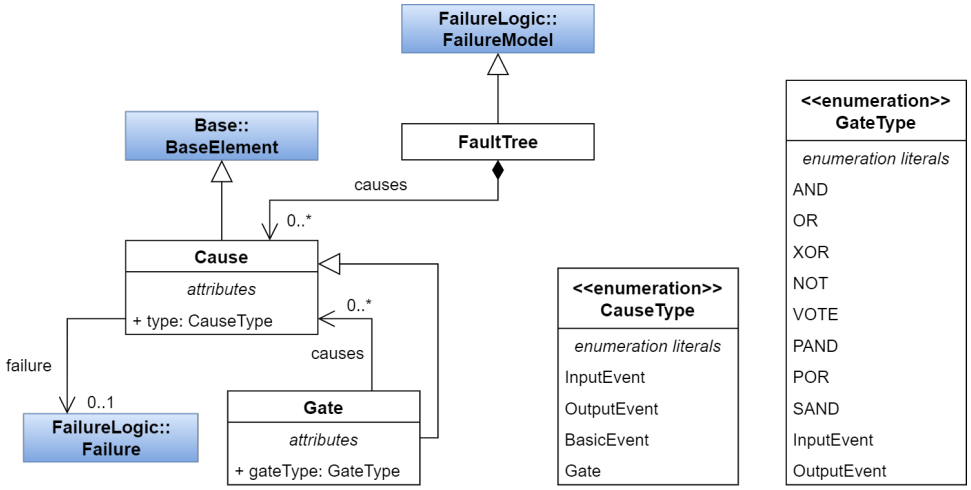
4.7 FailureLogic Sub-packages

The **ODE::FailureLogic::FTA** sub-package (Figure 17) captures the information produced during Fault Tree Analysis. *FaulTree* is a *FailureModel* subtype comprising a set of *Cause* elements. A *Cause* element can be an *input*, *output*, or a *basic event*, or a fault tree Gate. *Cause* references a *Failure*. A *Gate* is a *Boolean* logic *event connector*, which is a subtype of *Cause*, used to chain hierarchies of causes in a fault tree.

The **ODE::FailureLogic::FMEA** sub-package (Figure 18) captures the information produced during Failure Modes and Effects Analysis (FMEA). The *FMEA* element is a *Failure Model* comprising a set of entries (*FMEAEntry*). Each *FMEAEntry* relates an *output failure* from the SoS or CS elements under analysis with its SoS or component level effects. The FMEA element can be an FMEA or an FMEDA element. An FMEDA element has a set of *FMEDAEntries*. *FMEDAEntry* specializes *FMEAEntry* with a *diagnosisRate* double typed attribute, and a relationship with *ODE::FailureLogic::ProbDist*, used for calculating the SoS or CS unavailability for the referenced failure mode.

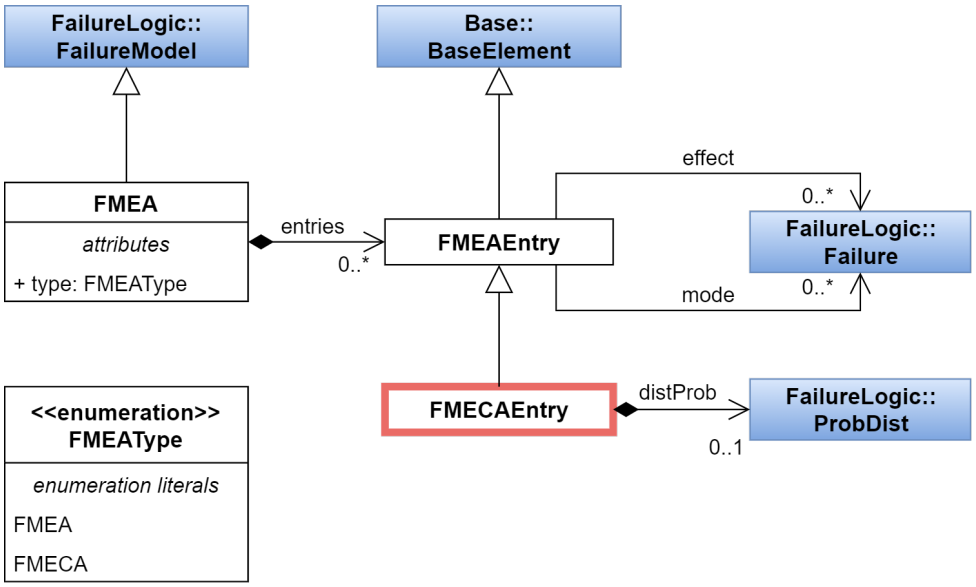
The **ODE::FailureLogic::Markov** sub-package provides the elements to support the

Figure 17 – FTA Package.



Source: Elaborated by the author.

Figure 18 – FMEA Package.



Source: Elaborated by the author.

analysis of dynamic and temporal aspects of the system/systems of systems behavior from the Markov analysis technique. A Markov model comprises a set of normal and failing states, one set as the initial state, and probabilistic transitions between states. A *State* references a *Failure* element from *ODE::FailureLogic* package. A *Transition* element represents a transition from a source state to a destination state. Each state *Transition* has a probabilistic attribute, calculated using a probability distribution (*ProbDist*) model. The introduction of *Fault*, *InternalFault*, *ExternalFault*, and *Error* failure subtypes and their relationships to the *ODE::FailureLogic* package may contribute the detail the causal relationships between failures and their origin into SoS and systems FTA, FMEA, and Markov models.

4.8 Final Considerations

This chapter introduced the extensions to the ODE meta-model to guide practitioners to perform Systems of Systems architectural design and safety analysis using state-of-the-art Model-Based Design languages like UML (OMG, 2017b), SysML (OMG, 2017a), MATLAB Simulink (MathWorks, 2021), CHESSML (MAZZINI *et al.*, 2016a), AADL (SAE, 2017), and EAST-ADL (EAST-ADL Association, 2013), and Model-Based Safety Assessment frameworks, including HiP-HOPS (PAPADOPOULOS *et al.*, 2011), CHESS-FLA (GALLINA; SEFER; REFSDAL, 2014), CHESS-SBA (MONTECCHI; GALLINA, 2017), AADL Error Annex (SAE, 2017), and AltaRica (BATTEUX; PROSVIRNOVA; RAUZY, 2018). In this thesis, the proposed meta-model was instantiated using MATLAB Simulink to support the model-based design of a Traffic Light Assistant Systems of Systems (KURAL *et al.*, 2014), and the HiP-HOPS integration with Simulink was used to support safety analysis (see Chapter 6). The proposed meta-model provides the basis for structuring a Systems of Systems model-based safety analysis process introduced in the next chapter

CHAPTER

5

SOSSAFE: COMPOSITIONAL SYSTEMS-OF-SYSTEMS SAFETY ANALYSIS

5.1 Initial Considerations

To answer RQ_3, this chapter introduces SoSSafe, a compositional approach to support safety analysis in SoS architectures. SoSSafe encompasses the definition of SoS safety analysis scope, the identification of combinations among system failure events that may lead the SoS to fail and the classification of their risks to the overall SoS safety, and the synthesis of fault trees to analyze how system failures propagate throughout the CSs and FMEA tables to identify the most critical components.

SoSSafe covers the whole safety life-cycle (see Section 2.3.2). The SoS preliminary architecture is the input to SoSSafe. Figure 19 depicts the approach phases and their relationships. The phases are as follows:

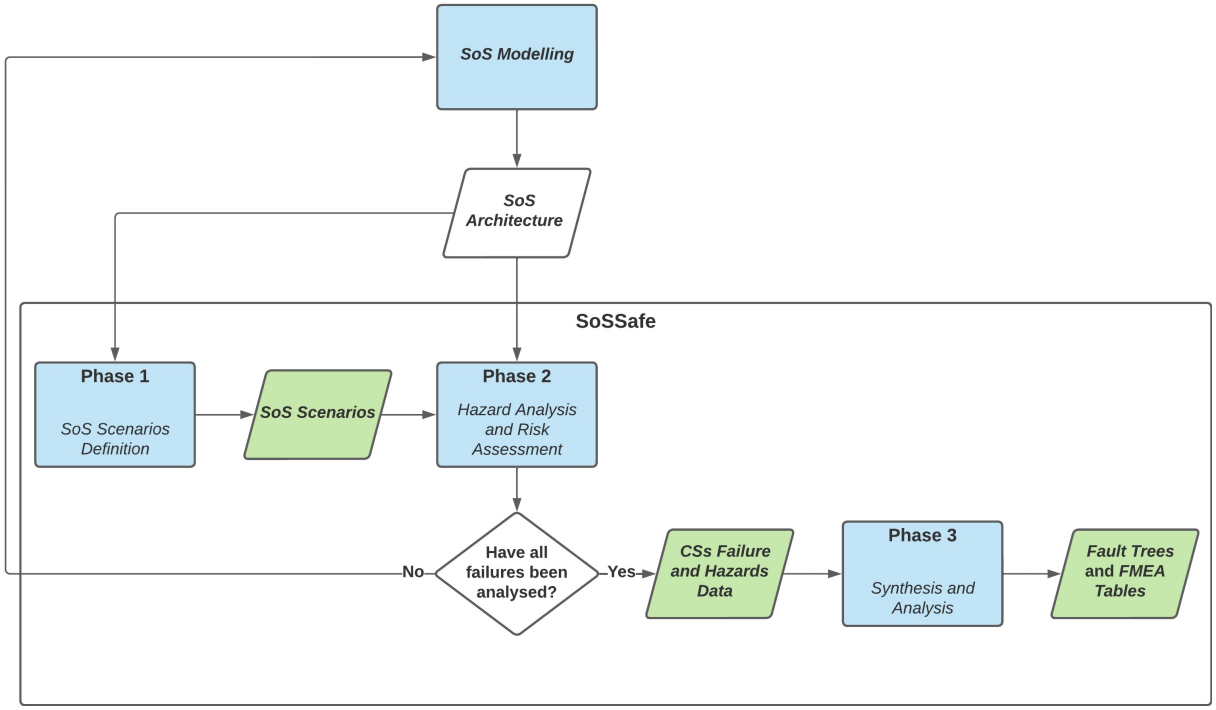
Phase 1 – Scenarios Definition: Definition of scenarios to be considered during SoS safety analysis;

Phase 2 – SoS HARA: It supports identifying combinations of CSs that may lead the SoS to fail and classifying the associated risks. It also defines CS and component level Failure Analysis to identify how CSs and components can fail and contribute to the occurrence of SoS hazards that could result in some harm to the SoS environment;

Phase 3 – Synthesis and Analysis: The safety assets (fault trees and FMEA tables) are generated for the SoS hazards identified.

SoSSafe phases can be performed iteratively and incrementally as long as additional information about the SoS architecture becomes available. SoS-level fault trees and FMEA

Figure 19 – SoSSafe phases and their relationships.



Source: Elaborated by the author.

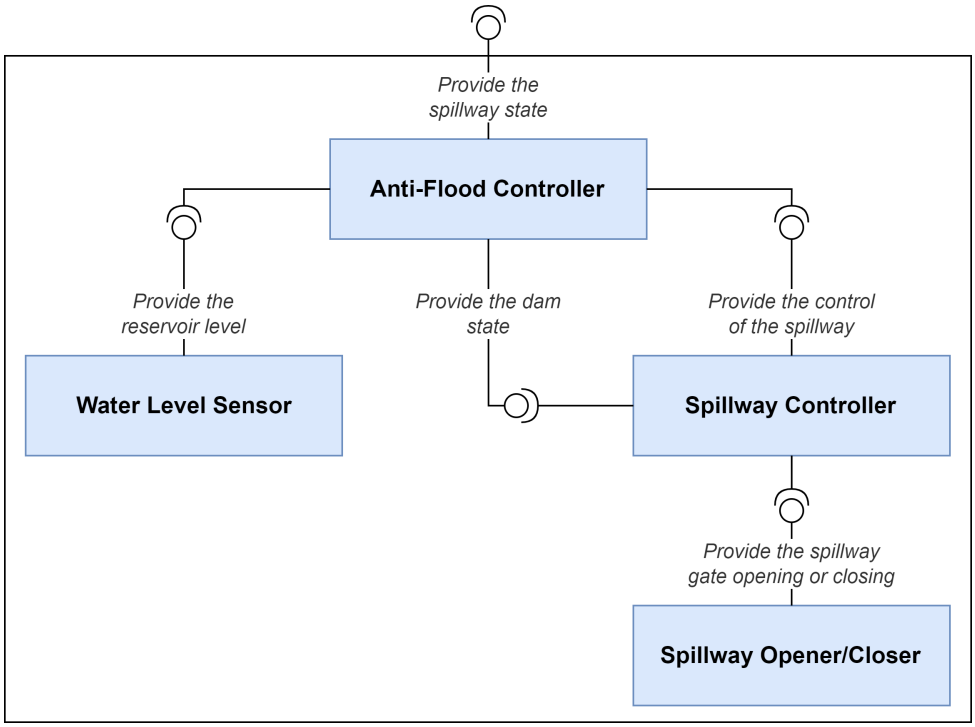
tables are outputs of SoSSafe to provide feedback information to the SoS Modeling phase. The analysis of fault trees and FMEA tables makes it possible to determine the changes needed in the SoS architectural design to address SoS safety goals and requirements. Fault trees are suitable to identify how SoS hazard effects propagate throughout CSs. FMEA tables are derived from the fault trees and help identify how CSs can fail directly, i.e., when a failure in a CS will lead to the occurrence of an SoS hazard, or indirectly, i.e., when a failure CS, in conjunction with one or more failures in other CSs, will cause an SoS hazard. Therefore, FTA and FMEA results can support SoS designers in identifying highly critical CSs or components that contribute to the occurrence of one or more hazardous events that may cause harm to the SoS environment and hence taking architectural decisions concerning the most suitable CSs to be incorporated into the SoS.

This chapter is organized as follows: Section 5.2 characterizes the SoS architecture, which is the input to SoSSafe. The following sections present the SoSSafe phases: Section 5.3 describes the *SoS Scenarios Definition* phase, Section 5.4 illustrates the *Hazard Analysis and Risk Assessment* phase, and Section 5.5 presents the *Synthesis and Analysis* phase. Finally, Section 5.6 highlights the final considerations.

5.2 SoSSafe Input

The SoS architectural model is the input of SoSSafe, represented, in this work, by an internal block diagram. This diagram captures block elements' internal structure in terms of their properties (Ports and Parts) and the connections between those properties. In this work, a block can be a system or a component. It is important to highlight that this work is not focused on proposing a novel approach to model system architectures. Other existing modeling techniques, such as UML Component Diagram (OMG, 2017b), Systems Modeling Language (SysML) (OMG, 2017a) and Service-Oriented Architecture Modeling Language (SOA-ML) (OMG, 2012) can be used to support the design of SoS architectures.

Figure 20 – AFS Functional Application Design.



Source: Elaborated by the author.

An SoS architecture model example from the hydroelectric power plant domain, called Anti-Flood System (AFS), is used throughout this chapter to describe the SoSSafe phases. The AFS architecture model is described through an UML Internal Block Diagram, as shown in Figure 21. AFS supports planning and controlling the spillway of a hydroelectric reservoir built upon a dam, which opens the spillway when the reservoir is almost complete (dam unsafe state). Otherwise, AFS keeps the spillway closed (safe dam state). AFS is considered a safety-critical system since potential failures related to AFS may cause a dam crash leading to catastrophic consequences, such as electricity shortage or even floods in cities around.

Figure 20 shows a high-level description of AFS components and their relationships and provided services through a component diagram, whereas Table 7 describes such components. In

Table 7 – AFS Components Description.

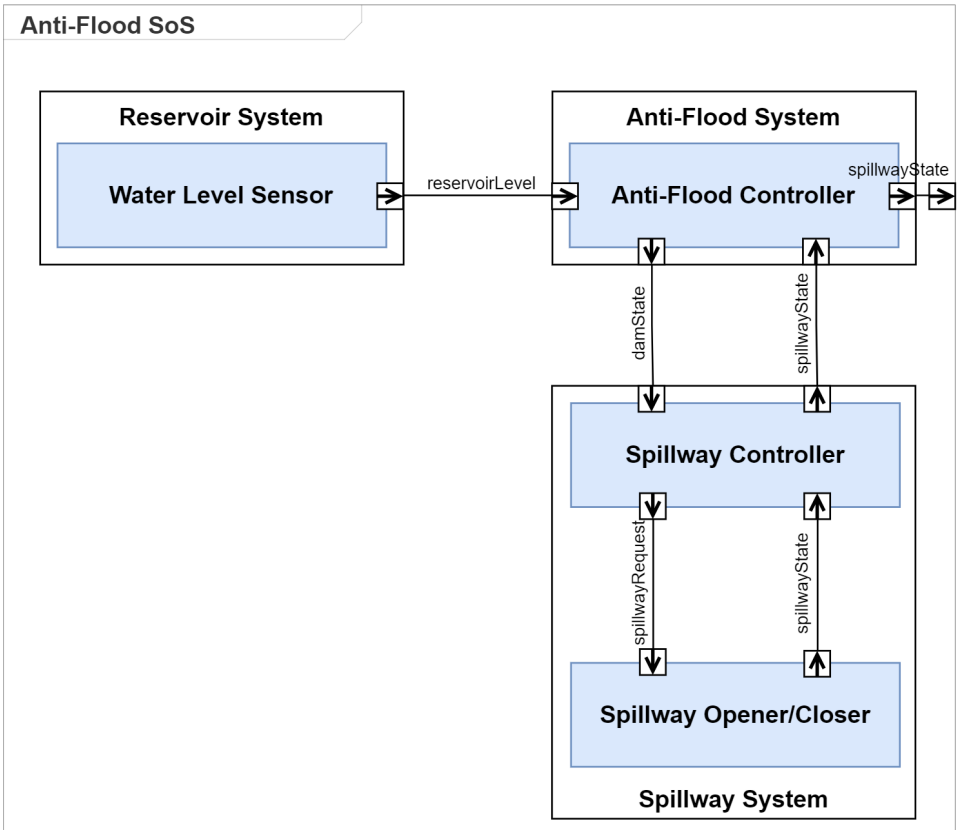
ID	Component	Description	Parameters	Return
1	Water Level Sensor	Provide the reservoir level: it returns the reservoir level in meters		*float* reservoirLevel
2	Anti-Flood Controller	Provide the dam state: it returns *true* if the dam is safe and *false* if it is unsafe (imminent flooding)	*float* reservoirLevel	*bool* damState
3	Spillway Controller	Provide the spillway state: it returns "true" if the spillway is open and "false" if it is closed	*bool* spillwayState	*bool* spillwayState
4	Spillway Controller	Provide the control of the spillway: it is open if the dam state is *unsafe* and closed when it is *safe*	*bool* damState	*bool* spillwayState
5	Spillway Opener/-Closer	Provide the spillway gate opening or closing: it is provided when requested		*bool* spillwayState

Source: Elaborated by the author.

this example, components are described as nouns (e.g., anti-flood controller), and the services implemented by such components are described as actions (e.g., *provide the spillway state*). AFS comprises four components: (i) *Water Level Sensor*; (ii) *Anti-Flood Controller*; (iii) *Spillway Controller*; and (iv) *Spillway Opener/Closer*. *Water Level Sensor* sends the reservoir level through the *Provide the reservoir level* service to the *Anti-Flood Controller*. *Anti-Flood Controller* determines the dam state through the *Provide the dam state*, and it sends the data to the *Spillway Controller*. The *Spillway Opener/Closer* provides the *Spillway gate opening or closing* service to the *Spillway Controller*. This service is activated by the *Spillway Controller* to open the spillway when the dam is in an unsafe state, and the spillway is closed to avoid the dam crash or to close the spillway when the dam is in a safe state, and the spillway is still opened.

After defining the components and their relationships, they must be allocated to AFS CSs. The AFS components have been allocated into three CSs, as shown in Figure 21 that presents the AFS architecture. Such CSs are:

Figure 21 – AFS Internal Block Diagram.



Source: Elaborated by the author.

Reservoir System: It returns the value of the reservoir water level;

Spillway System: It controls the spillway opening or closing

Anti-Flood System: It unifies the two CSs presented before since it is responsible for the spillway planning and controlling.

After defining the SoS architecture, the first phase of SoSSafe can be started.

5.3 Phase 1 – SoS Scenarios Definition

The *SoS Scenarios Definition* phase intends to define the scenarios at the SoS-level. This phase is important because defining the SoS scenarios provides the scope of SoSSafe to identify the potential threats to the SoS safety, classify the associated risks, establish the safety goals, and allocate them to the CSs. The input of this phase is the SoS architecture defined before.

To define the scenarios, reliable information about the application domain is needed. For each project, domain experts discuss the possible scenarios to reach a consensus. To facilitate the scenario documentation and its dissemination throughout the project team, SoSSafe proposes

a template that could be followed to document each scenario. This template comprises the following fields:

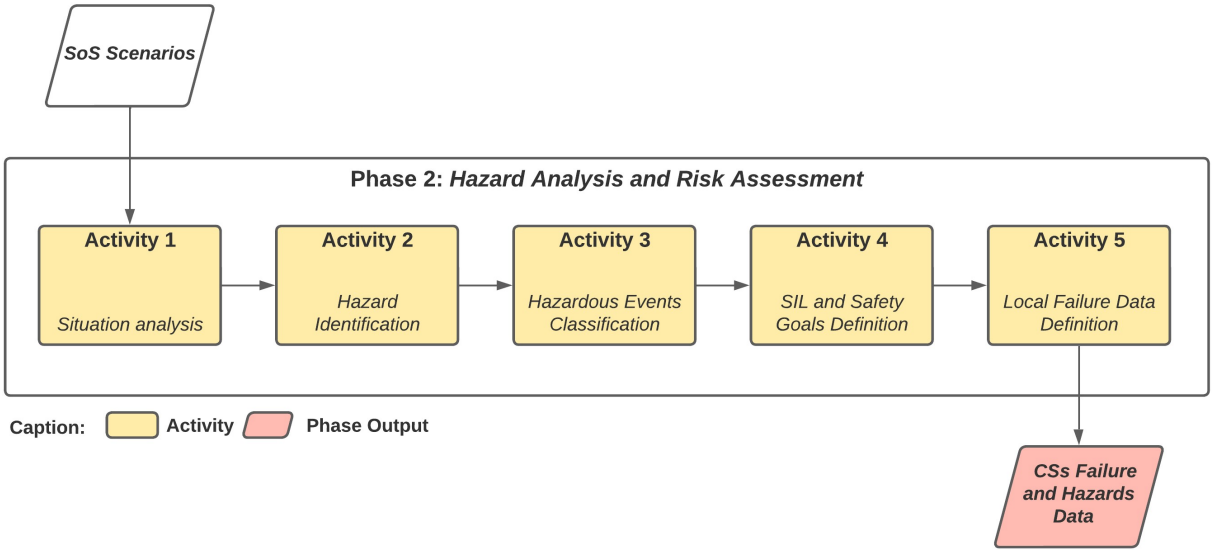
Scenario ID - Scenario Name: a unique identifier and a title that briefly describes the scenario are provided;

Description: Brief description of the scenario in terms of the interactions between CSs that provide a given emergent behavior or behaviors to be further considered during the remainder of the SoSSafe. For instance, in AFS, this scenario is identified: "the scenario where the spillway is opened to preserve the structure of the damn" (MORI *et al.*, 2018).

Based on the SoS scenarios, the next phase can be started.

5.4 Phase 2 – Hazard Analysis and Risk Assessment

Figure 22 – HARA Activities.



Source: Elaborated by the author.

In this phase, the input is the SoS architecture and the SoS scenarios. SoS HARA intends to identify and analyze the SoS hazards and their associated risks and annotate the CSs and components with local failure data. This phase is important because SoS hazards identification and analysis, as well as annotations of the CSs and components with local failure data provides relevant information to automate the generation of safety work products, such as fault trees and FMEA tables. These can lead the project team to make decisions concerning the SoS architecture and, hence, achieve an acceptable SoS safety level. As shown in Figure 22, this phase is divided into the following activities:

Activity 1 – Situation Analysis: It aims to define the SoS operational situations. The input is the SoS scenarios, and the output is the SoS operational situations;

Activity 2 – Hazard Identification: It aims to identify the SoS hazards. The input is the SoS architecture and the SoS scenarios and the output is the SoS hazards classified through the Redmond taxonomy;

Activity 3 – Hazardous Events Classification: It aims to classify hazardous events. The input is the SoS architecture, SoS hazards, and the operational situations. The output is the risk attributes for each hazardous event;

Activity 4 – SIL and Safety Goals Definition: It aims to define the SILs for each hazardous event. The input is the SoS architecture, SoS hazards, and the operational situations. The output is the SILs for each hazardous event;

Activity 5 – Local Failure Data Definition: Finally, the CSs and components are annotated with local failure data definition. The input is the SoS architecture, SoS hazards, and the operational situations. The output is the CSs and components local failure data.

These activities, together with examples, are described in more detail as follows.

5.4.1 Situation Analysis

Before starting the hazard identification, the operation situations in which SoS malfunctioning behaviour will result in a hazardous event shall be determined. The operation situations must be described, both for cases where the SoS is correctly used and others where it is incorrectly used in a foreseeable way. In AFS, the operation situations considered are *low energy generation* and *high energy generation*.

5.4.2 Hazard Identification

In this activity, the hazards at the SoS-level are identified. Considering the SoS scenarios, the SoS architecture is analyzed to identify possible combinations of failures in different CSs that can lead to a failure in providing emerging behavior and, consequently, any damages to the SoS environment that includes property, people, or organizations.

The SoS hazards are represented through the logical combination of CS outputs failures (output deviations). The generic syntax adopted to represent output deviations is the following: *GuideWord-ConstituentSystem.PortName*. Such syntax consists of three parts. The first part (*GuideWord*) is the HAZOP guide word described in Chapter 2, the second part (*ConstituentSystem*) is the CS name, and the third one (*PortName*) is the port where the failure will be propagated. The first line of Table 8 is an example of an SoS hazard. In this case, the hazard occurs due to an omission of the *Anti-Flood System* CS. Otherwise, the hazard could occur because of the

Table 8 – Hazard Example.

Failure Expression	Omission-ReservoirSystem.spillwayState
Hazard Type	Interface
Likelihood	S3
Severity	L3
SIL	D
Safety Goal	The spillway must be open when the dam state is unsafe

Source: Elaborated by the author.

omission of two or more services, in which the failure expressions are combined through logical operators, such as *AND*, *OR*, or *NOT*. One example is the following hazard that occurs when the Anti-Flood System does not provide at least one of its outputs (*damState* and *spillwayState*) when requested: *Omission-AntiFloodSystem.damState OR Omission-AntiFloodSystem.spillwayState*. Besides that, the Redmond taxonomy must be considered to assess the *Hazard Type* attribute. In the hazard example shown in Table 8, the hazard type defined is *Interface*. After defining the SoS hazards, the hazardous events are classified in the next activity.

5.4.3 Hazardous Events Classification

In this activity, the hazardous events are classified through risk attributes. This activity relies on the safety standards under consideration to define the risk attributes and the ways to assess them. Regarding the hazard example shown in Table 8, the columns *likelihood* and *severity* represent the standard-related attributes. It is important to highlight that the same hazard has different risk attributes values, since each hazard may be associated with one or more operational situations. Therefore, the risk attribute values of Table 8 are from the hazardous event with the most stringent SIL associated with this hazard. The next activity defines the most stringent hazardous event for each hazard.

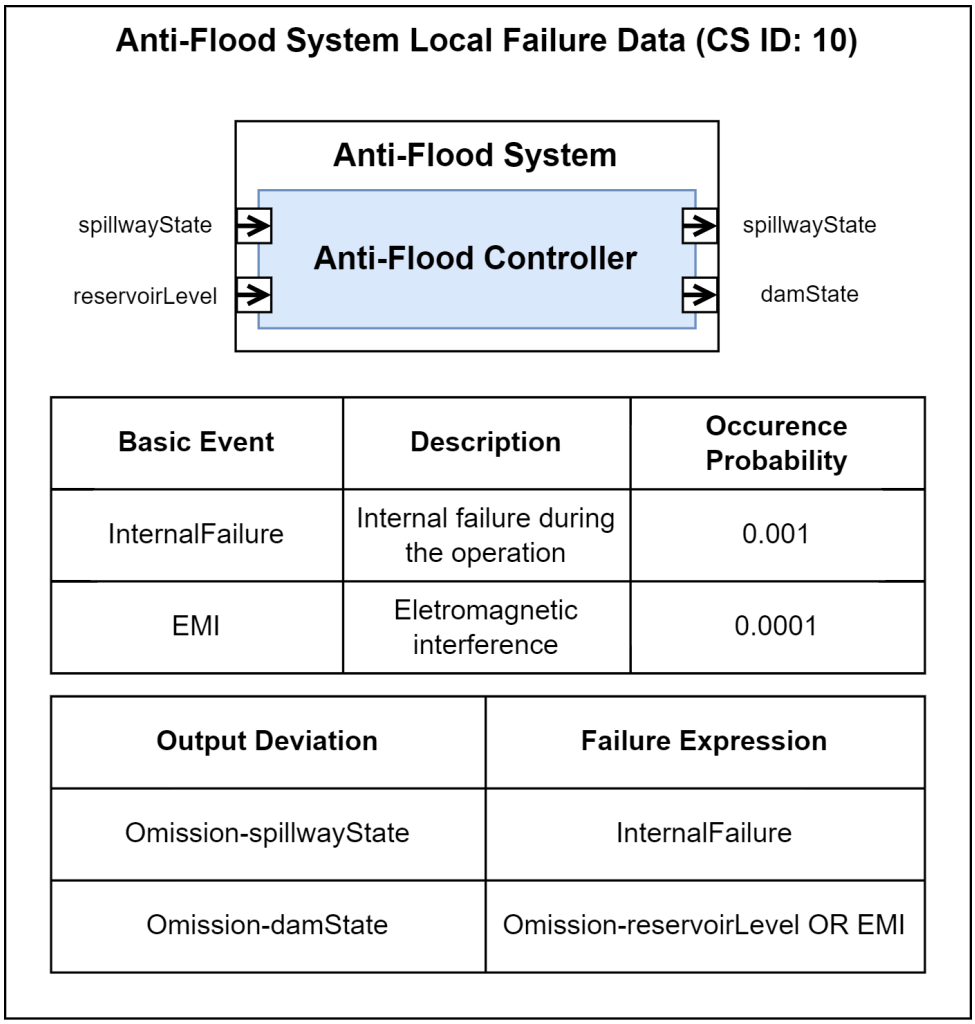
5.4.4 SIL and Safety Goals Definition

This activity determines a SIL for each hazardous event using the risk parameters. The way to assess the SIL relies on the safety standard. About the hazard in Table 8, there are two hazardous events associated since there are two operational situations (low and high energy generation). The risk attributes of the first hazardous event have the values S2 and L2, and the second one has the values S3 and L3 for the attributes *severity* and *likelihood*, respectively. Using a safety standard, the hazardous events have SIL *B* and *D*, respectively. Since SIL *D* is more stringent than SIL *B*, the SIL of the hazard in Table 8 is B, and the risk attributes are S3 and L3. Besides, the SIL definition is also needed to define the safety goal for the SoS hazard. In the

example, the safety goal is *the spillway must be open when the dam state is unsafe*. The next activity defines the local failure data for each CS and component.

5.4.5 Local Failure Data Definition

Figure 23 – Anti-Flood System Local Failure Data.



Basic Event	Description	Occurence Probability
InternalFailure	Internal failure during the operation	0.001
EMI	Eletromagnetic interference	0.0001

Output Deviation	Failure Expression
Omission-spillwayState	InternalFailure
Omission-damState	Omission-reservoirLevel OR EMI

Source: Elaborated by the author.

In this activity, each CS and component in the SoS architecture is annotated with its local failure data that describes how that CS or component can fail and how it responds to failures propagated from other CS or components in the SoS. The local failure data takes the form of a set of failure expressions relating failures at a CS's outputs (output deviations) to a logical combination of internal failure modes (basic events) and failures received at the CS's inputs (input deviations). The syntax and set of HAZOP guide words for input and output deviations specification are the same used to describe SoS hazards. Figure 23 shows an example of local failure data for the CS *Anti-Flood System* with the following data:

- **CS Name and ID:** It identifies the CS through its name and unique identification. In the example, the CS name is *Anti-Flood System* and the CS ID is equal to *10*;
- **Input/Output Ports:** The CS ports are named to define the failure expressions. In the example, the input ports are *spillwayState* and *reservoirLevel* and the output ones are *spillwayState* and *damState*;
- **Basic Events:** Here, the failure modes' names, description, and probability are described. In the example, the basic events are internal failure (*InternalFailure*) and electromagnetic interference (*EMI*);
- **Output Deviation and its Failure Expression:** Finally, it describes the output deviation and the logical combination of one more or input deviations that may cause the output deviation. In the example, there are two output deviations: *Omission-spillwayState*, whose cause is an internal failure (*InternalFailure*), and *Omission-damState*, whose cause is an omission of *reservoirLevel* input port or electromagnetic interference (*Omission-reservoirLevel OR EMI*).

The FTA and FMEA are performed in the next phase after all failures have been analyzed. If all failures have not been analyzed yet, SoS modeling should be reviewed, as shown in Figure 19.

5.5 Phase 3 – Synthesis and Analysis

In this phase, the inputs are the SoS hazard info and the CS local failure data. Such input is used to generate a network of interconnected fault trees defining the relationships between failures of system outputs and their root causes in the failure modes of individual CSs. After that, the minimal cut sets are obtained and used as a basis to perform FMEA, which directly relates individual CS failures to their effects on the rest of the SoS. This phase is performed automatically through safety analysis tools. The output of this phase is the fault trees, the respective minimal cut sets, and the FMEA tables. In the next chapter, a fault tree example is presented.

5.6 Final Considerations

This chapter presented the process that intends to help SoS practitioners to assure SoS safety. To demonstrate its efficacy, SoSSafe was performed in a real illustrative case from the automotive domain, and the results obtained are exposed in the next chapter.

CHAPTER

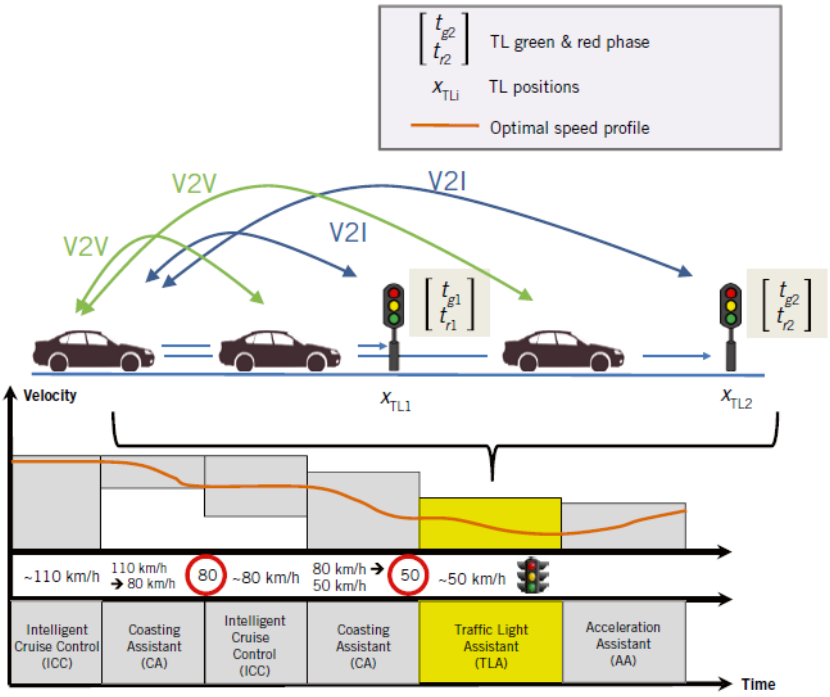
6

SOSSAFE ILLUSTRATIVE CASE

6.1 Initial Considerations

To answer RQ_3, this chapter presents the SoSSafe application through an illustrative study from the automotive domain. Section 6.2 describes the SoS considered in the illustrative study. Section 6.3, Section 6.4 and Section 6.5 illustrate the application of SoSSafe. Finally, Section 6.6 presents the final considerations.

Figure 24 – Traffic Light Assistant Overview.



Source: Reich *et al.* (2020).

6.2 Traffic Light Assistant Illustrative Study Description

This section presents the SoS that has been considered to evaluate the applicability of SoSSafe. The SoS used in this work is a Traffic Light Assistant (TLA), a Cooperative Autonomous System (CAS) that automatically optimizes the longitudinal vehicle speed with the SoS (named Ego Vehicle (EV)) to avoid unnecessary stops at red traffic lights, as depicted in Figure 24. TLA decreases the number of times the EV should be accelerated or decelerated, thus minimizing energy loss due to unnecessary braking at red lights. Besides, by reducing waiting time and introducing smoother speed profiles, significant improvements can occur, such as an increased lifetime of components, relaxed production quality requirements, and improved mobility (REICH; SCHNEIDER, 2018; KURAL *et al.*, 2014).

TLA is a safety-critical SoS since catastrophic failures can occur, such as vehicle crashes or vehicle collisions with pedestrians. For instance, complete knowledge about the road conditions and traffic light signal phasing to perform its mission may not be accessible in some scenarios. If the current state of a traffic light or a pedestrian at the side of a crossing is observed by an onboard camera, the information would only be available to a vehicle close by the pedestrian. If such a vehicle is equipped with automated driving functions, it will adapt its velocity accordingly to stop in front of the traffic light or to let the pedestrian pass safely. However, the following vehicle might not directly see the pedestrian and, therefore, would be unable to anticipate the behavior of the preceding vehicle (Digital Dependability Identities and the Open Dependability Exchange Meta-Model). In this scenario, people may get hurt or even die in severe situations. Therefore, systems safety analysis can help avoid hazardous situations. Before starting SoSSafe application, TLA architecture has been proposed in the following since it is the input of the approach.

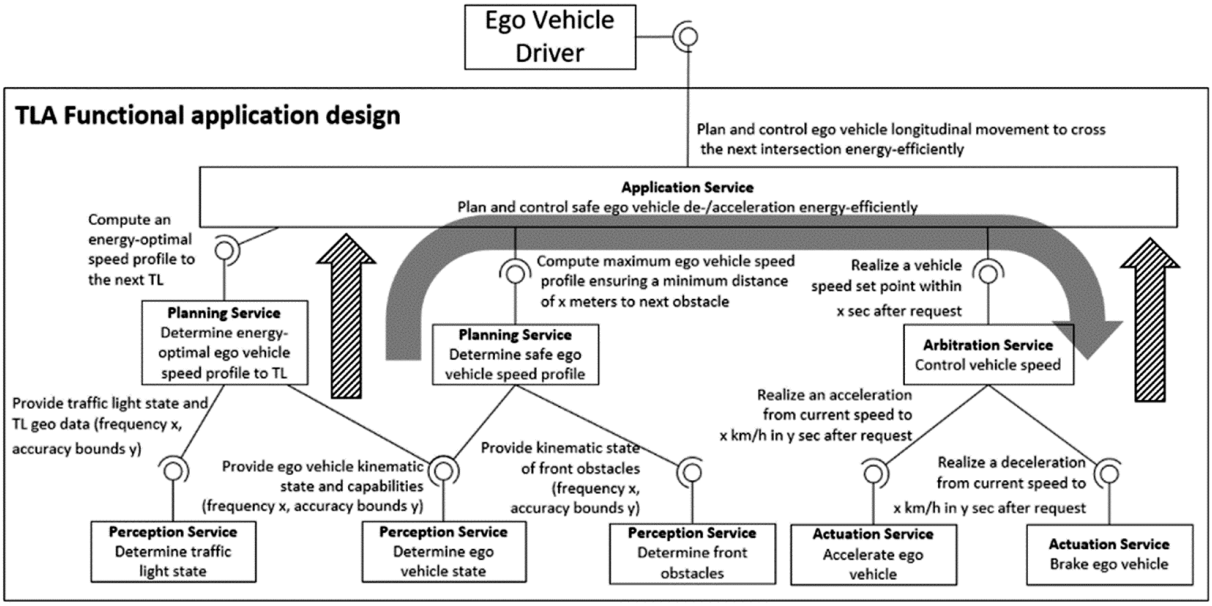
6.2.1 Traffic Light Assistant Functional Application Design

Figure 25 shows the functional application design of the TLA that describes its services and the relationship between them. Such services have been grouped into five types of service:

Perception Service: It measures environment and infrastructure data. The *Determine traffic light state* service uses a V2I technology to determine the traffic light state and an onboard satellite navigation system with relatively good data quality to determine the traffic light position. The *Determine ego vehicle state* service provides relevant EV data, such as its speed, acceleration, and position. The *Determine front obstacles* service uses Vehicle-to-Vehicle (V2V) technology and onboard radar to define front vehicles;

Planning Service: In the planning service, an energy-optimal profile speed to the next traffic light is provided by the *Determine energy-optimal ego vehicle speed profile to TL* service.

Figure 25 – Traffic Light Assistant Functional Application Design.



Source: Reich and Schneider (2018).

In addition, the optimal safe speed profile that ensures a safe distance between the EV and the front vehicle is provided by the *Determine safe ego vehicle speed profile* service;

Application Service: It is the set of services delivered to the EV. Here, the *Plan and control safe ego vehicle de-/acceleration energy-efficiently* service provides the automated control of longitudinal movement to cross intersections efficiently;

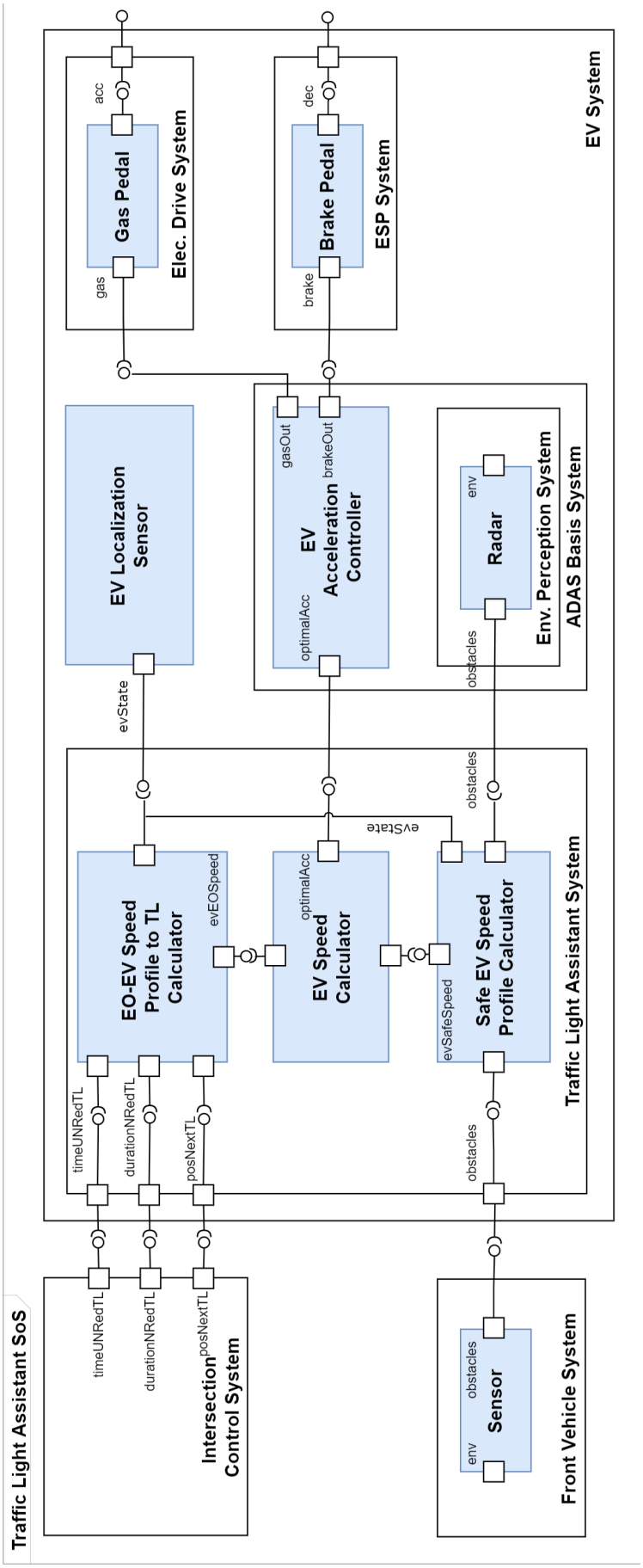
Arbitration Service: Through the speed profile obtained by the planning services, the *Control vehicle speed* service determines if the EV needs to increase or decrease its speed;

Actuation Service: Finally, the EV is accelerated or decelerated by its actuators. If the EV needs to be accelerated, the *Accelerate ego vehicle* service is responsible for soliciting the actuator to accelerate the car from the current speed to x Km/h. In turn, the *Brake ego vehicle* service sends a requisition to the actuator to brake the car from the current speed to x Km/h.

6.2.2 TLA Architecture and Data Flow

After defining the services and the relationship between them, the services exposed in Figure 25 have been associated with the CSs that compose the SoS. As shown in Figure 26, a component diagram has been proposed, where the light blue boxes represent the components and the white ones, that represent the CSs, end with the word *system*. The components and CSs of the TLA component diagram are responsible for implementing the services described in the TLA functional application design. The CSs and their components are described as follows:

Figure 26 – Traffic Light Assistant Component Diagram.



Source: Elaborated by the author.

Intersection Control System: It intends to determine the traffic light states of each intersection through V2I communication;

Front Vehicle System: It intends to detect vehicles that can be in front of the EV. Here, the EV perceives front vehicles through V2V communication since the EV receives data emitted by a sensor from the front vehicle;

EV System: On-boarding the EV, it intends to determine the ideal vehicle state to minimize EV energy consumption. EV system has been broken down into four CSs as follows:

Traffic Light Assistant System: Here, EV acceleration and deceleration regarding safe driving and energy-saving are defined. This CSs is composed of three components: *EO-EV Speed Profile to TL Calculator* (it computes the energy-optimal speed profile to the next traffic light), *Safe EV Speed Profile Calculator* (it computes maximum EV speed profile, ensuring a minimum distance of x meters to next obstacle), and *EV Speed Calculator* (it plans and controls EV acceleration or deceleration regarding the safety and the EV energy-saving);

ADAS Basis System: This CS is based on Advanced Driver-Assistance System (ADAS) that intends to detect objects (including vehicles) that can be in front of the EV, as well as to define if the EV should be accelerated or not through the *EV Acceleration Controller* component. The *Environment Perception System* CS has the same goal of the *Front Vehicle System* CS. However, front obstacles are perceived through an EV onboard radar;

Electric Drive System and Electronic Stability Program (ESP) System: Here, the EV is accelerated through the *Gas Pedal* from the *Electric Drive System* CS or decelerated as a result of the *Brake Pedal* component from the *ESP System* CS. The decision of accelerating or braking the vehicle is provided by the *Acceleration Control* CS from the *ADAS Basis System* CS.

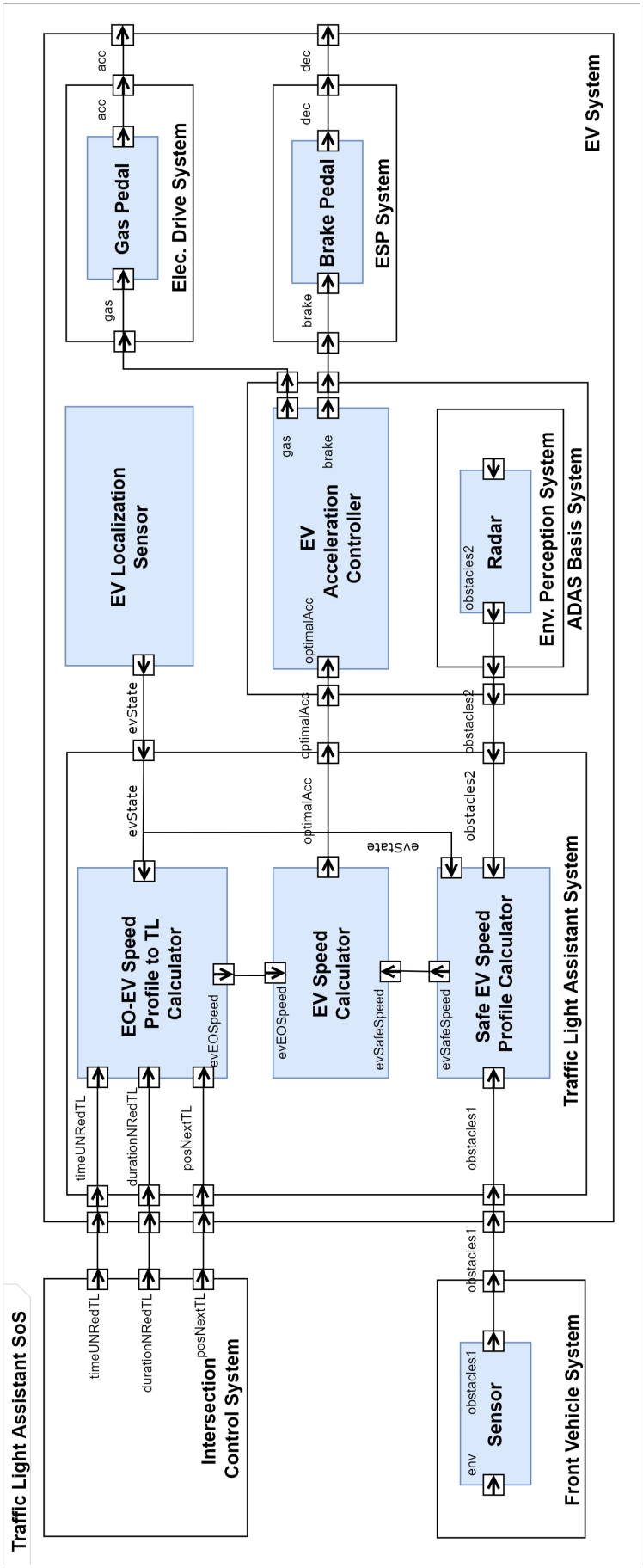
To define the hazards and failure modes in the next phase, the TLA internal block diagram that defines the data flow between the CSs and components of the SoS is presented in Figure 27. Based on the TLA internal block diagram, which can be used as input, the SoSSafe application is started.

6.3 Phase 1 – Scenarios Definition

In this illustrative study, the following scenario has been considered:

Scenario Title: Driving EV Towards Intersections.

Figure 27 – Traffic Light Assistant Internal Block Diagram.



Source: Elaborated by the author.

Scenario Description: In the Driving EV Towards Intersections scenario, TLA conducts the EV to the next traffic light to cross the intersection.

When the EV crosses intersections at the traffic light red phase, hazardous events can occur, since TLA failures can hurt or even kill people or cause financial damage to the environment, such as car crashes and infrastructure damage to the city. Therefore, this scenario must be considered to be analyzed in the following phases. To restrict the scope of this illustrative study, only hazards that could cause damage to citizens have been considered in the analysis.

6.4 Phase 2 – SoS HARA

In this section, SoS HARA applied to TLA is described.

6.4.1 Situation Analysis

In this illustrative study, TLA hazards were analyzed under three operational situations:

Low-Speed : Situation in which TLA is operated under EV low speed (less than 30 Km/h);

Medium Speed: Situation in which TLA is operated under EV medium speed (more than 30 Km/h and less than 60 Km/h);

High-Speed : Situation in which TLA is operated under EV high speed (more than 60 Km/h).

Hazards can occur in both operational situations, but they could occur under other ones regarding nature conditions, such as foggy days and snowfall, or traffic conditions, such as low traffic, medium traffic, and high traffic. The following activity identifies and analyzes potential hazards at the SoS level taking into account the operational situations defined.

6.4.2 Hazard Identification

Analyzing TLA architecture and the scenario considered in this illustrative case, hazards can occur due to the combination of EV System failures. An output failure of the EV System can cause hazards. If omission output failures occur, EV can be accelerated or decelerated and then cross the intersection when the traffic light is in a red phase. The same can occur in the case of early, late, and incorrect output failures since the vehicle can accelerate or decelerate earlier (early output failure) or later (late output failure) than expected more or less than needed (incorrect output failure). The hazards identified are described as follows:

Hazard Name (H_1): EV is not accelerated when expected

Failure Expression: Omission-EVSystem.ElecDriveSystem.acc

Context: The EV is crossing the intersection, but it needs to be accelerated. In case of acceleration omission, the EV can get stopped in the intersection during the red phase

Hazard Name (H_2): EV is not decelerated when expected

Failure Expression: Omission-EVSystem.ESPSystem.dec

Context: If this situation occurs, the EV will not stop at a red phase when reaching the traffic light and then cross the intersection

Hazard Name (H_3): EV is accelerated too early

Failure Expression: Early-EVSystem.ElecDriveSystem.acc

Context: The EV is stopped at the intersection, and the next command is to accelerate the EV three seconds after the next green phase. If the command is executed too early (before the next green phase), the EV can cross the intersection at the red phase

Hazard Name (H_4): EV is accelerated too late

Failure Expression: Early-EVSystem.ESPSystem.dec

Context: The EV is stopped at the intersection, and the next command is to accelerate the EV three seconds after the next green phase, and the next red phase will be in 10 seconds. If the command is executed too late (more than 13 seconds), the EV will cross the intersection at the red phase

Hazard Name (H_5): EV is decelerated too late

Failure Expression: Late-EVSystem.ESPSystem.dec

Context: The EV is being conducted between two intersections, and there is a front obstacle. TLA calculates that the EV should be decelerated in 3 seconds. However, the command starts after the recommended time. If it occurs, a collision with the front obstacle can occur.

Besides hazards described before, others could be considered, such as incorrect values in which the EV is accelerated or decelerated more or less than the needed, or commission failures. Only the hazards defined before have been considered in the next activity to restrict the scope of this illustrative study.

Table 9 – Severity Classes.

Severity	**S0**	**S1**	**S2**	**S3**
Description	No injuries	Light and Moderate Injuries	Severe and life-threatening injuries (survival is probable)	Life-threatening injuries (survival is uncertain), fatal injuries

Source: ISO (2011).

Table 10 – Probability of Exposure Classes.

Severity	**E0**	**E1**	**E2**	**E3**	**E4**
Description	Incredible	Very low probability	Low probability	Medium probability	High probability

Source: ISO (2011).

Table 11 – Controllability Classes.

Severity	**C0**	**C1**	**C2**	**C3**
Description	Controllable in general	Simply controllable	Normally controllable	Difficult to control or uncontrollable

Source: ISO (2011).

6.4.3 Hazardous Events Classification

To proceed with hazardous events classification, ISO 26262 has been used because the industry has adopted this standard to develop automotive electric systems like TLA (ISO, 2011). The hazardous events have been classified through the following attributes:

Type: Defined through Redmond taxonomy to classify SoS hazards;

Severity: Table 9 depicts the classes of severity proposed in ISO 26262;

Probability of Exposure: Table 10 shows the probability of exposure regarding operational situations proposed in ISO 26262;

Controllability: Table 11 describes the classes of controllability proposed in ISO 26262.

The results are exposed in Table 12. Regarding the type, all hazards have been considered as interface hazards in the Redmond taxonomy since they are caused by sharing faulty data between the CSs and components through a defined channel.

Regarding severity, all hazards have been considered as S3 severity. H_1 because fatal injuries can occur in case of the vehicle is at high speed and suddenly get stuck on the road. Therefore, a car crash between the EV and the vehicle behind it can occur. H_2 because if the EV crosses the intersection at high speed, a fatal crash can occur. H_3 and H_4 because if the EV crosses the intersection and the vehicle crossing the same intersection at the same time are on high speed, a fatal crash can occur. Finally, H_5 because car crashes involving vehicles on high speed can cause fatal injuries.

Regarding the probability of exposure, E4 has been considered because the EV stops in more than 10% of the operation time (ISO, 2011). For controllability, C3 has been considered for all hazards because TLA is an autonomous system. Therefore, humans do not have control of TLA and then cannot do anything to avoid hazards. From severity, probability of exposure, and controllability attributes, SILs have been defined as follows.

Table 12 – Hazardous Events Classification and SIL results.

Hazard	**Type**	**Severity**	**Probability of Exposure**	**Controllability**	**ASIL**
H_1	Resource	S3	E0	C3	D
H_2	Resource	S3	E0	C3	D
H_3	Resource	S3	E0	C3	D
H_4	Resource	S3	E0	C3	D
H_5	Resource	S3	E0	C3	D

Source: Elaborated by the author.

6.4.4 SIL and Safety Goals Definition

As explained in Chapter 2, SIL is known as ASIL in ISO 26262. Table 13 depicts all ASILs defined in ISO 26262 in which each hazardous event can be classified into five ASILs: Quality Management (QM), A, B, C, or D. QM refers to the standard's consideration that it is below ASIL A in which there is no safety relevance, and only standard QM processes are required to fulfill any relevant requirements. In turn, ASIL A indicates the least critical level, and D indicates the most critical level (ISO, 2011). As can be seen in Table 13, all hazards can be considered as ASIL D. The results are shown in the sixth column of Table 12. Regarding the safety goals, they are defined through the safety goal for each hazard identified. The way to define is the same as for CS hazards. In the next phase, CS and component local failure data definition are described, since it is important to determine the causes of each SoS hazard identified.

Table 13 – ASIL Determination.

Severity Class	**Probability Class**	**Controllability Class**		
		C1	C2	C3
S1	E1	QM	QM	QM
S1	E2	QM	QM	QM
S1	E3	QM	QM	A
S1	E4	QM	A	B
S2	E1	QM	QM	QM
S2	E2	QM	QM	A
S2	E3	QM	A	B
S2	E4	A	B	C
S3	E1	QM	QM	A
S3	E2	QM	A	B
S3	E3	A	B	C
S3	E4	B	C	D

Source: ISO (2011).

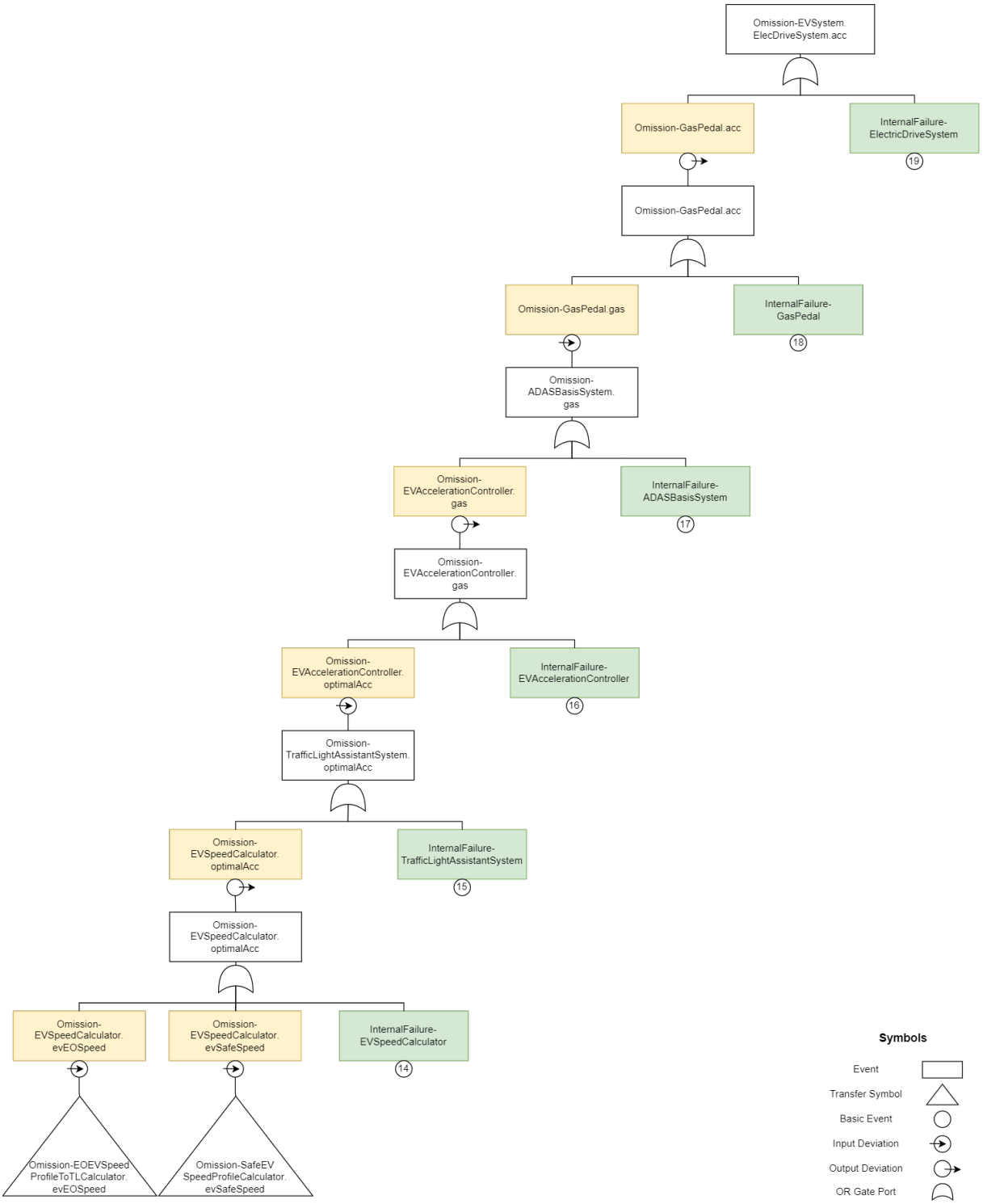
6.4.5 Local Failure Data Definition

In this phase, CS and component local failure data are defined. As SoSSafe is a top-down approach, the analysis begins at the CS level and then at the component level. To simplify the illustration of SoSSafe applicability, only the hazards caused by omission output failures will be considered in this activity and in the next phase. The output deviation for each CS and component output port are shown in Appendix A. In this illustrative study, the cause of each output deviation is the combination of omission of input ports or internal failure. In the next phase, all data gathered in this phase will be synthesized to generate the SoSSafe outputs for H_1 hazard.

6.5 Phase 3 – Synthesis and Analysis

Finally, FTA can be performed to generate the fault tree for the H_1 hazard. Figure 28, Figure 29, and Figure 30 show the fault trees generated. The triangle is used as a matter of convenience to avoid extensive duplication in a fault tree or to allow a large tree to be represented on a number or smaller trees for clarity. In this case, the fault tree flow has been broken into more parts to be represented in another fault tree (in this case, such parts are represented in Figure 29 and Figure 30). The circles with an arrow denote an input deviation where the arrow enters the circle and an output deviation where the arrow leaves the circle. The green boxes with a circle

Figure 28 – H_1 Fault Tree – Part 1.



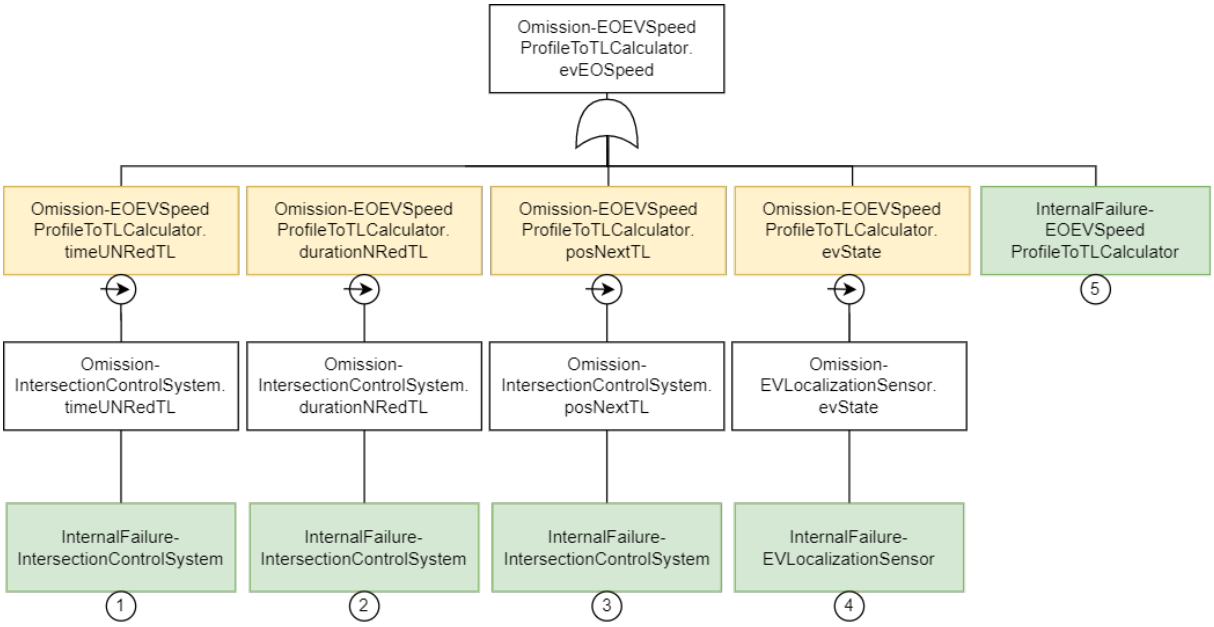
Source: Elaborated by the author.

Table 14 – H_1 FMEA Table.

Component/System failure	**Direct effects on the system**	**Effects caused in conjunction with other events**
1	ElecDriveSystem	–
2	ElecDriveSystem	–
3	ElecDriveSystem	–
4	ElecDriveSystem	–
5	ElecDriveSystem	–
6	ElecDriveSystem	–
7	ElecDriveSystem	–
8	ElecDriveSystem	–
9	ElecDriveSystem	–
10	ElecDriveSystem	–
11	ElecDriveSystem	–
12	ElecDriveSystem	–
13	ElecDriveSystem	–
14	ElecDriveSystem	–
15	ElecDriveSystem	–
16	ElecDriveSystem	–
17	ElecDriveSystem	–
18	ElecDriveSystem	–
19	ElecDriveSystem	–

Source: Elaborated by the author.

Figure 29 – H_1 Fault Tree – Part 2.



Source: Elaborated by the author.

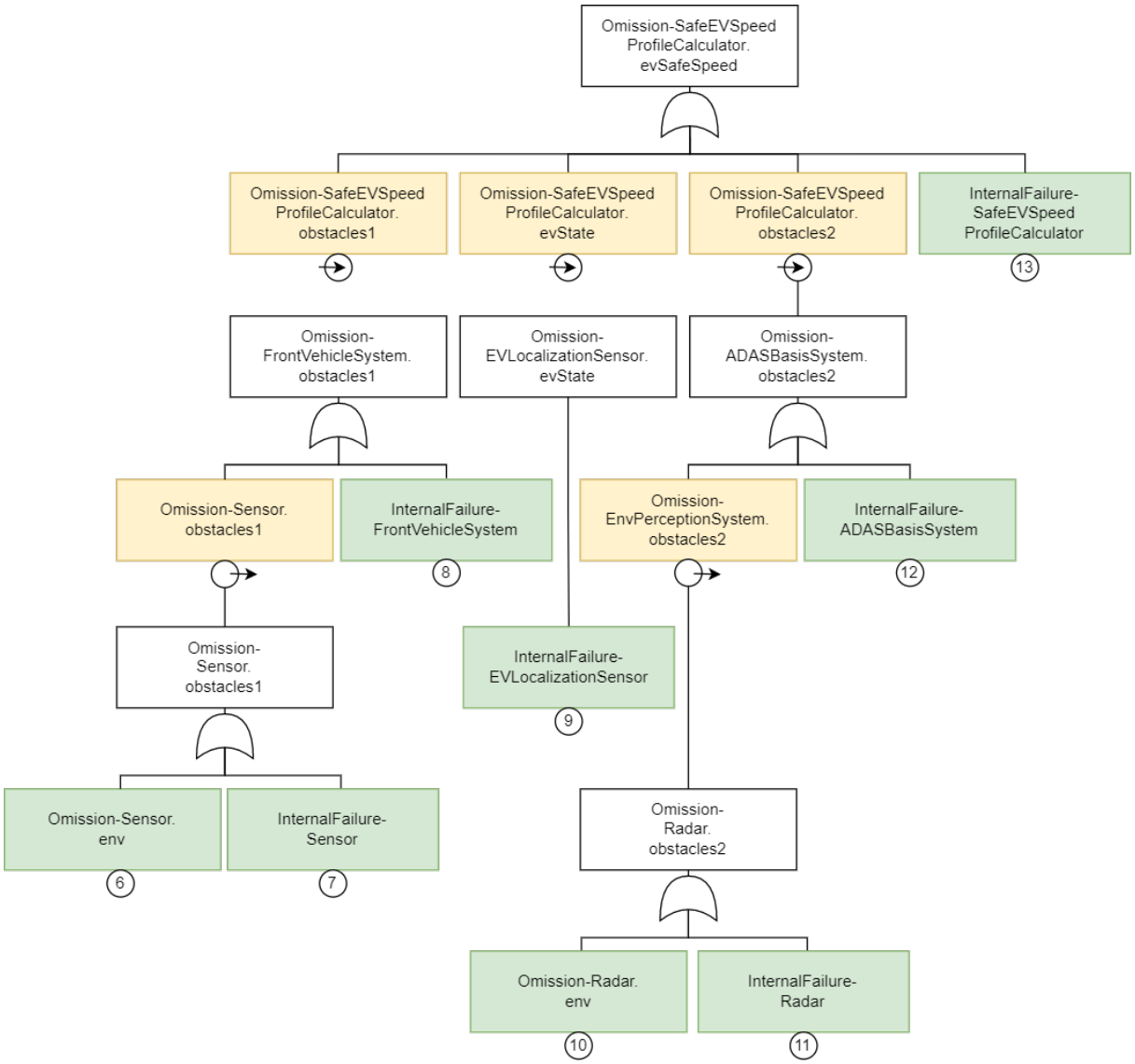
represent basic initiating fault event that requires no further development, and the numbers inside the circles under the green boxes are basic events' unique identifiers used to generate the FMEA table.

Table 14 presents the FMEA table generated from H_1 fault tree. The first column represents the basic events' unique identifiers. The second column indicates if such failures directly affect the system. As the fault tree generated is composed of only *OR* operators, all component or system failures can trigger *Elec.DriveSystem* failures without the necessity of occurring in conjunction with other basic events.

6.6 Final Considerations

As described in this chapter, the SoSSafe approach helps manage SoS hazards since the outputs generated by the approach help to decide which CSs should integrate the SoS. The illustrative case shown in this chapter can be considered a preliminary evaluation of SoSSafe, and a complete validation is left as future work. The next chapter presents the conclusions of this work.

Figure 30 – H_1 Fault Tree – Part 3.



Source: Elaborated by the author.

CHAPTER

7

CONCLUSIONS AND FUTURE WORK

7.1 Initial Considerations

This chapter presents a summary of contributions obtained from the research conducted in this work (Section 7.2), the limitations found throughout the development of this work (Section 7.3), and, finally, the future research suggestions to extend the SoS safety analysis state-of-art (Section 7.4).

7.2 Contributions

In this work, the general contribution is to support SoS safety analysis to define which CSs meet the systems' safety properties to be incorporated into the SoS operation. Such contribution can be divided into the following specific contributions:

Systematic Mapping: In this contribution, a systematic mapping was conducted in which several studies regarding risk management practices were analyzed. The mapping results identified system safety gaps in the current state-of-art literature focusing on SoS. The full version of this systematic mapping has been published in ICSA-C (LOPES *et al.*, 2020);

SoSSafe Meta-Model: SoSSafe meta-model intends to support SoS design and safety analysis. SoSSafe meta-model proposes a structured way to model the information regarding SoS and its CSs to perform safety analysis at SoS and CS levels at design time to avoid hazards during SoS operation. Such meta-model supports the identification and analysis of SoS hazards by incorporating a specific taxonomy to classify them proposed by Redmond, Michael and Shebalin (2008), as well as the identification and analysis of output deviations and their causes that can be failure modes or input deviations at the CS level;

SoSSafe Approach: This approach intends to semi-automate the process of analyzing SoS safety properties and estimating the effects of CSs failures in SoS architectures. The output of SoSSafe can be helpful to guide safety analysts and the whole team to define which CSs should incorporate the SoS and hence decrease the number of system failures during SoS operation.

7.3 Limitations

The most significant limitation faced in this work is the difficulty of applying SoSSafe on a realistic SoS. The illustrative study exposed in this work has been based on a real SoS from the automotive domain. However, the data gathered is from a few papers published. Therefore, relevant information regarding the SoS could not be found since the papers found do not have in-depth information, and the authors of the papers are not available to provide more information about the SoS.

Still, regarding the validation process, this work has not followed a structured way to validate the SoSSafe such as a case study validation process. Besides that, SoSSafe approach has not been compared with other works that address safety in complex systems. Finally, SoSSafe has not been validated in different application domains to guarantee that the approach can be helpful for different application domains.

7.4 Future Research

The following areas have been identified as a subject of future research:

Systematic Mapping: An update of the systematic mapping performed in this work intends to define new SoS safety gaps;

SoSSafe Validation: Regarding the validation limitations, further efforts intend to validate the SoSSafe approach and meta-model in other realistic case studies from different application domains, with the support of domain experts. Besides that, it is intended to compare SoSSafe approach and meta-model with related works to support systems safety analysis;

SoSSafe Extension: Since safety and cybersecurity are related dependability attributes, cybersecurity threats can trigger hazards. Therefore, it is recommended that both properties should be analyzed together. In this context, it is intended to extend the SoSSafe approach and meta-model to support cybersecurity threats.

BIBLIOGRAPHY

AITKEN, J. M.; ALEXANDER, R.; KELLY, T. A case for dynamic risk assessment in nec systems of systems. In: IEEE. **2010 5th International Conference on System of Systems Engineering**. [S.l.], 2010. p. 1–6. Citations on pages 58, 59, 62, and 65.

____. A risk modelling approach for a communicating system of systems. In: IEEE. **2011 IEEE International Systems Conference**. [S.l.], 2011. p. 442–447. Citations on pages 58, 62, and 65.

ALEXANDER, R.; HALL-MAY, M.; KELLY, T. Characterisation of systems of systems failures. In: CITESEER. **Proceedings of the 22nd International System Safety Conference**. [S.l.], 2004. Citations on pages 26 and 28.

ALEXANDER, R.; KELLY, T. Hazard analysis through simulation for systems of systems. In: **Proceedings of the 24th International Systems Safety Conference**. [S.l.: s.n.], 2006. Citation on page 28.

AVIZIENIS, A.; LAPRIE, J.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. **IEEE Transactions on Dependable and Secure Computing**, v. 1, n. 1, p. 11–33, Jan 2004. Citations on pages 36 and 37.

AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. **IEEE transactions on dependable and secure computing**, IEEE, v. 1, n. 1, p. 11–33, 2004. Citations on pages 61 and 62.

AXELSSON, J.; KOBETSKI, A. Towards a risk analysis method for systems-of-systems based on systems thinking. In: IEEE. **2018 Annual IEEE International Systems Conference (SysCon)**. [S.l.], 2018. p. 1–8. Citations on pages 58, 62, and 65.

____. Towards a risk analysis method for systems-of-systems based on systems thinking. In: **2018 Annual IEEE International Systems Conference (SysCon)**. [S.l.: s.n.], 2018. p. 1–8. ISSN 2472-9647. Citations on pages 63 and 65.

BATTEUX, M.; PROSVIRNOVA, T.; RAUZY, A. Enhancement of the altarica 3.0 stepwise simulator by introducing an abstract notion of time. In: **Safety and Reliability–Safe Societies in a Changing World**. [S.l.]: CRC Press, 2018. p. 915–921. Citation on page 79.

BAUMGART, S.; FRÖBERG, J.; PUNNEKKAT, S. Analyzing hazards in system-of-systems: Described in a quarry site automation context. In: IEEE. **2017 Annual IEEE International Systems Conference (SysCon)**. [S.l.], 2017. p. 1–8. Citations on pages 40, 57, 58, 62, 63, 64, and 65.

____. A process to support safety analysis for a system-of-systems. In: IEEE. **2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)**. [S.l.], 2020. p. 61–66. Citation on page 28.

BOARDMAN, J.; SAUSER, B. System of systems - the meaning of of. In: **2006 IEEE/SMC International Conference on System of Systems Engineering**. [S.l.: s.n.], 2006. p. 6 pp.–. Citations on pages 28 and 32.

BOURQUE, P.; FAIRLEY, R. E. **Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0**. [S.l.]: IEEE Computer Society Press, 2014. Citation on page 32.

BOUTI, A.; KADI, D. A. A state-of-the-art review of fmea/fmeca. **International Journal of reliability, quality and safety engineering**, World Scientific, v. 1, n. 04, p. 515–543, 1994. Citation on page 44.

ČAUŠEVIĆ, A. A risk and threat assessment approaches overview in autonomous systems of systems. In: IEEE. **2017 XXVI International Conference on Information, Communication and Automation Technologies (ICAT)**. [S.l.], 2017. p. 1–6. Citations on pages 58, 62, and 65.

CAVALCANTE, E.; CACHO, N.; LOPES, F.; BATISTA, T. Challenges to the development of smart city systems: A system-of-systems view. In: **Proceedings of the 31st Brazilian Symposium on Software Engineering**. [S.l.: s.n.], 2017. p. 244–249. Citation on page 25.

CHEN, P.; UNEWISSE, M. Sos thinking: an approach to conceptualising and understanding military systems-of-systems. **International Journal of System of Systems Engineering**, v. 8, p. 74, 01 2017. Citations on pages 25 and 32.

CMU. **Open Source AADL Tool Environment (OSATE)**. 2020. Accessed on 19.01.2021. Available: <osate.org>. Citation on page 67.

COMMITTEE, S. I. S.-. *et al.* Arp4761 guidelines and methods for conducting the safety assessment process on civil airborne system and equipment. **Warrendale, Pennsylvania: Society of Automotive Engineers**, 1996. Citation on page 45.

CONROW, E. H. Risk management for systems of systems. **CrossTalk**, v. 18, n. 2, p. 8–12, 2005. Citations on pages 17, 53, 57, 58, 59, 60, 62, 63, and 65.

DAHMANN, J. S.; BALDWIN, K. J. Understanding the current state of us defense systems of systems and the implications for systems engineering. In: IEEE. **2008 2nd Annual IEEE Systems Conference**. [S.l.], 2008. p. 1–7. Citation on page 33.

DAHMANN, J. S.; JR, G. R.; LANE, J. **Systems engineering for capabilities**. [S.l.], 2008. Citations on pages 33, 62, and 65.

DARAMOLA, O.; STåLHANE, T.; SINDRE, G.; OMORONYIA, I. Enabling hazard identification from requirements and reuse-oriented hazop analysis. In: **2011 4th International Workshop on Managing Requirements Knowledge**. [S.l.: s.n.], 2011. p. 3–11. ISSN null. Citation on page 41.

DEIS. **Open Dependability Exchange (ODE) Profile V2**. [S.l.], 2020. Available: <https://www.deis-project.eu>. Citations on pages 67, 71, and 73.

DELANGE, J. **Architecture Analysis and Design Language. Number SAE AS5506C. SAE International**. 2016. Accessed on 12.09.2021. Available: <https://www.sae.org/standards/content/as5506/>. Citation on page 75.

DELANGE, J.; FEILER, P. Architecture fault modeling with the aadl error-model annex. In: **2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications**. [S.l.: s.n.], 2014. p. 361–368. Citations on pages 26, 27, and 48.

DEZFULI, H.; BENJAMIN, A.; EVERETT, C.; SMITH, C.; STAMATELATOS, M.; YOUNGBLOOD, R. Nasa system safety handbook. **NASA/SP-2010–580, Washington, DC: NASA**, v. 1, 2011. Citation on page 26.

DIESTE, O.; PADUA, A. G. Developing search strategies for detecting relevant experiments for systematic reviews. In: IEEE. **First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)**. [S.l.], 2007. p. 215–224. Citation on page 53.

DUNJÓ, J.; FTHENAKIS, V.; VÍLCHEZ, J. A.; ARNALDOS, J. Hazard and operability (hazop) analysis. a literature review. **Journal of hazardous materials**, Elsevier, v. 173, n. 1-3, p. 19–32, 2010. Citations on pages 39 and 41.

DYBA, T.; DINGSOYR, T.; HANSSEN, G. K. Applying systematic reviews to diverse study types: An experience report. In: IEEE. **First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)**. [S.l.], 2007. p. 225–234. Citation on page 66.

EAST-ADL Association. **EAST-ADL Domain Model Specification version V2.1.12**. 2013. Available: <https://www.east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf>. Citation on page 79.

Eclipse Foundation. **Eclipse Papyrus Modeling environment**. 2017. Available: <https://www.eclipse.org/papyrus/>. Citation on page 67.

____. **Eclipse Modeling Framework platform**. 2022. Available: <https://www.eclipse.org/modeling/emf/>. Citation on page 68.

ELSHENAWY, M.; ABDULHAI, B.; EL-DARIEBY, M. Towards a service-oriented cyber–physical systems of systems for smart city mobility applications. **Future Generation Computer Systems**, v. 79, p. 575 – 587, 2018. ISSN 0167-739X. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X17307471>. Citations on pages 25 and 32.

EUROCAE. Arp4754 – guidelines for development of civil aircraft and systems. **SAE International**, 2010. Citations on pages 29, 38, 39, 40, and 43.

FENELON, P.; MCDERMID, J. A. New directions in software safety: Causal modelling as an aid to integration. In: **Workshop on Safety Case Construction, York (March 1994)**. [S.l.: s.n.], 1992. Citations on pages 46 and 47.

____. An integrated tool set for software safety analysis. **Journal of Systems and Software**, v. 21, n. 3, p. 279 – 290, 1993. ISSN 0164-1212. Applying Specification, Verification, and Validation Techniques to Industrial Software Systems. Available: <http://www.sciencedirect.com/science/article/pii/016412129390029W>. Citation on page 46.

GALLINA, B.; JAVED, M. A.; MURAM, F. U.; PUNNEKKAT, S. A model-driven dependability analysis method for component-based architectures. In: **Proceedings - 38th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2012**. [S.l.: s.n.], 2012. p. 233–240. ISBN 9780769547909. Citation on page 75.

GALLINA, B.; SEFER, E.; REFSDAL, A. Towards safety risk assessment of socio-technical systems via failure logic analysis. In: IEEE. **International Symposium on Software Reliability Engineering Workshops (ISSRE)**. [S.l.], 2014. p. 287–292. Citations on pages 67 and 79.

GANDHI, S. J.; GOROD, A.; SAUSER, B. A systemic approach to managing risks of sos. In: **2011 IEEE International Systems Conference**. [S.l.: s.n.], 2011. p. 412–416. Citation on page 32.

____. A systemic approach to managing risks of sos. **IEEE Aerospace and Electronic Systems Magazine**, IEEE, v. 27, n. 5, p. 23–27, 2012. Citations on pages 57, 58, 59, 62, 63, and 65.

GE, X.; PAIGE, R. F.; MCDERMID, J. A. Probabilistic failure propagation and transformation analysis. In: SPRINGER. **International Conference on Computer Safety, Reliability, and Security**. [S.l.], 2009. p. 215–228. Citation on page 47.

GOROD, A.; SAUSER, B.; BOARDMAN, J. System-of-systems engineering management: A review of modern history and a path forward. **IEEE Systems Journal**, IEEE, v. 2, n. 4, p. 484–499, 2008. Citations on pages 32, 62, and 65.

GREENWOOD, D.; SOMMERVILLE, I. Responsibility modeling for identifying sociotechnical threats to the dependability of coalitions of systems. In: IEEE. **2011 6th International Conference on System of Systems Engineering**. [S.l.], 2011. p. 173–178. Citations on pages 58, 61, 62, 63, and 66.

____. Responsibility modeling for the sociotechnical risk analysis of coalitions of systems. In: IEEE. **2011 IEEE International Conference on Systems, Man, and Cybernetics**. [S.l.], 2011. p. 1256–1261. Citations on pages 58, 62, 63, and 66.

GUARINIELLO, C.; MOCKUS, L.; RAZ, A. K.; DELAURENTIS, D. A. Towards intelligent architecting of aerospace system-of-systems. In: **2019 IEEE Aerospace Conference**. [S.l.: s.n.], 2019. p. 1–11. Citations on pages 25 and 32.

GUNAWAN, I.; GOROD, A.; HALLO, L.; NGUYEN, T. Developing a system of systems management framework for the fukushima daiichi nuclear disaster recovery. In: **2017 International Conference on System Science and Engineering (ICSSE)**. [S.l.: s.n.], 2017. p. 563–568. Citations on pages 25 and 32.

HABLI, I. **Model-based assurance of safety-critical product lines**. Phd Thesis (PhD Thesis) — University of York, 2009. Citation on page 39.

HAIMES, Y. Y. Modeling complex systems of systems with phantom system models. **Systems Engineering**, Wiley Online Library, v. 15, n. 3, p. 333–346, 2012. Citations on pages 58, 59, 62, and 65.

____. Systems-based guiding principles for risk modeling, planning, assessment, management, and communication. **Risk Analysis: An International Journal**, Wiley Online Library, v. 32, n. 9, p. 1451–1467, 2012. Citations on pages 53, 58, 59, 62, and 65.

____. Risk modeling of interdependent complex systems of systems: Theory and practice. **Risk Analysis**, Wiley Online Library, v. 38, n. 1, p. 84–98, 2017. Citations on pages 58, 60, 62, and 65.

HANSEN, K. M.; WELLS, L.; MAIER, T. Hazop analysis of uml-based software architecture descriptions of safety-critical systems. **Proceedings of NWUML**, p. 59–78, 2004. Citation on page 41.

IEC. IEC 61508 - Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related System. In: . [S.l.: s.n.], 2010. Citation on page 73.

ISO. Iso 31000: Risk management–principles and guidelines. **International Organization for Standardization**, 2009. Citation on page 59.

____. Iso 26262: Road vehicles-functional safety. **International Standard ISO/FDIS**, 2011. Citations on pages 29, 36, 37, 38, 39, 40, 45, 75, 99, 100, and 101.

____. Iso/iec/ieee international standard – systems and software engineering – system of systems (sos) considerations in life cycle stages of a system. **ISO/IEC/IEEE 21839:2019(E)**, p. 1–40, 2019. Citation on page 31.

JAMSHIDI, M. System of systems - innovations for 21st century. In: **Industrial and Information Systems. IEEE the Third international Conference on**. [S.l.: s.n.], 2008. p. 6–7. Citation on page 32.

KÄSSMEYER, M.; SCHULZE, M.; SCHURIUS, M. A process to support a systematic change impact analysis of variability and safety in automotive functions. In: ACM. **Proceedings of the 19th International Conference on Software Product Line**. [S.l.], 2015. p. 235–244. Citation on page 38.

KI-ARIES, D.; FAILY, S.; DOGAN, H.; WILLIAMS, C. Assessing system of systems security risk and requirements with oasosis. In: IEEE. **2018 IEEE 5th International Workshop on Evolving Security & Privacy Requirements Engineering (ESPRE)**. [S.l.], 2018. p. 14–20. Citations on pages 58, 62, 63, and 64.

KINDER, A.; HENSHAW, M.; SIEMIENIUCH, C. A model based approach to system of systems risk management. In: IEEE. **2015 10th System of Systems Engineering Conference (SoSE)**. [S.l.], 2015. p. 122–127. Citations on pages 58, 59, 62, 63, and 65.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [S.l.], 2007. Citations on pages 51, 53, and 66.

KLETZ, T. A. Hazop—past and future. **Reliability Engineering & System Safety**, Elsevier, v. 55, n. 3, p. 263–266, 1997. Citation on page 41.

KOBETSKI, A.; AXELSSON, J. Towards safe and secure systems of systems: Challenges and opportunities. In: **Proceedings of the Symposium on Applied Computing**. [S.l.: s.n.], 2017. p. 1803–1806. Citation on page 26.

KRISTEN, J. Systems engineering guide for systems of systems (version 1.0). **Systems and Software Engineering, Office of the Deputy Under Secretary of Defense for Acquisition and Technology**, 2008. Citations on pages 58 and 59.

KURAL, E.; JONES, S.; PARRILLA, A. F.; GRAUERS, A. Traffic light assistant system for optimized energy consumption in an electric vehicle. In: IEEE. **2014 International Conference on Connected Vehicles and Expo (ICCVE)**. [S.l.], 2014. p. 604–611. Citations on pages 79 and 92.

LANE, J. A. **What is a System of Systems and Why Should I Care?** 2013. Tecnical Report, University of Southern California. Citation on page 34.

LEITE, F. L.; SCHNEIDER, D.; ADLER, R. Dynamic risk management for cooperative autonomous medical cyber-physical systems. In: SPRINGER. **International Conference on Computer Safety, Reliability, and Security**. [S.l.], 2018. p. 126–138. Citations on pages 25, 32, 58, 62, 63, 64, and 65.

LEVESON, N. A new accident model for engineering safer systems. **Safety science**, Elsevier, v. 42, n. 4, p. 237–270, 2004. Citation on page 65.

LEVESON, N. G. Software safety: Why, what, and how. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 18, n. 2, p. 125–163, Jun. 1986. ISSN 0360-0300. Available: <http://doi.acm.org/10.1145/7474.7528>. Citation on page 37.

LISAGOR, O.; MCDERMID, J.; PUMFREY, D. Towards a practicable process for automated safety analysis. In: CITESEER. **24th International system safety conference**. [S.l.], 2006. v. 596, p. 607. Citations on pages 28 and 46.

LIU, H.-C.; LIU, L.; LIU, N. Risk evaluation approaches in failure mode and effects analysis: A literature review. **Expert Systems with Applications**, v. 40, n. 2, p. 828 – 838, 2013. ISSN 0957-4174. Available: <http://www.sciencedirect.com/science/article/pii/S0957417412009712>. Citations on pages 44 and 45.

LOCK, R. Developing a methodology to support the evolution of system of systems using risk analysis. **Systems Engineering**, Wiley Online Library, v. 15, n. 1, p. 62–73, 2012. Citations on pages 58, 59, 62, 64, and 65.

LOLLINI, P.; MORI, M.; BABU, A.; BOUCHENAK, S. Amadeos sysml profile for sos conceptual modeling. In: **Cyber-Physical Systems of Systems**. [S.l.]: Springer, 2016. p. 97–127. Citation on page 27.

LOPES, S. d. S.; VARGAS, I. G.; OLIVEIRA, A. L. de; BRAGA, R. T. V. Risk management for system of systems: A systematic mapping study. In: IEEE. **2020 IEEE International Conference on Software Architecture Companion (ICSA-C)**. [S.l.], 2020. p. 258–265. Citations on pages 51, 67, and 107.

LOPEZ, D. Lessons learned from the front lines of the aerospace industry - balancing complexity and risk. In: **Conference on System of Systems Engineering, IEEE/SMC International**. [S.l.: s.n.], 2006. p. 5–14. Citations on pages 25 and 31.

MAIER, M. W. Architecting principles for systems-of-systems. **Systems Engineering: The Journal of the International Council on Systems Engineering**, Wiley Online Library, v. 1, n. 4, p. 267–284, 1998. Citations on pages 25, 31, 32, and 33.

MALTA, M.; STRATHDEE, S. A.; GARCIA, P. J. The brazilian tragedy: Where patients living at the 'earth's lungs' die of asphyxia, and the fallacy of herd immunity is killing people. **EClinicalMedicine**, Elsevier, v. 32, 2021. Citation on page 25.

MATHEW, E. Intelligent transport systems and its challenges. In: HASSANIEN, A. E.; SHAALAN, K.; TOLBA, M. F. (Ed.). **Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2019**. Cham: Springer International Publishing, 2020. p. 663–672. Citations on pages 25 and 32.

MathWorks. **Simulink - Simulation and Model-Based Design**. 2021. Available: <https://www.mathworks.com/products/simulink.html>. Citations on pages 67 and 79.

MAZZINI, S.; FAVARO, J.; PURI, S.; BARACCHI, L. CHESS: An open source methodology and toolset for the development of critical systems. **CEUR Workshop Proceedings**, v. 1835, p. 59–66, 2016. ISSN 16130073. Citations on pages 67 and 79.

MAZZINI, S.; FAVARO, J. M.; PURI, S.; BARACCHI, L. Chess: an open source methodology and toolset for the development of critical systems. In: **EduSymp/OSS4MDE@ MoDELS**. [S.l.: s.n.], 2016. p. 59–66. Citations on pages 27 and 48.

MCDERMID, J. A.; NICHOLSON, M.; PUMFREY, D. J.; FENELON, P. Experience with the application of hazop to computer-based systems. In: IEEE. **COMPASS'95 Proceedings of the Tenth Annual Conference on Computer Assurance Systems Integrity, Software Safety and Process Security'**. [S.l.], 1995. p. 37–48. Citation on page 41.

MENGMENG, Z.; HONGHUI, C.; XIAOXUE, Z.; AIMIN, L.; JUNXIAN, L. Functionality evaluation of system of systems architecture based on extended influence diagrams. **Journal of Systems Engineering and Electronics**, v. 29, n. 3, p. 510–518, 2018. Citation on page 26.

Ministry of Defence. Hazop studies on systems containing programmable electronics. **Defence Standard 00-58 Issues 1 and 2**, v. 2, 2000. Citation on page 42.

MOD. **DEF-STAN 00-56 Issue 4 Part 1: Safety management requirements for defence systems**. [S.l.], 2007. Citations on pages 37 and 39.

MONTECCHI, L.; GALLINA, B. SafeConcert: a Metamodel for a Concerted Safety Modeling of Socio-Technical Systems. In: **5th International Symposium on Model-Based Safety and Assessment (IMBSA 2017)**. Trento, Italy: [s.n.], 2017. (LNCS, v. 10437), p. 129–144. Citations on pages 67 and 79.

MORI, M.; CECCARELLI, A.; LOLLINI, P.; FRÖMEL, B.; BRANCATI, F.; BONDAVALLI, A. Systems-of-systems modeling using a comprehensive viewpoint-based sysml profile. **Journal of Software: Evolution and Process**, Wiley Online Library, v. 30, n. 3, p. e1878, 2018. Citation on page 86.

NIELSEN, C. B.; LARSEN, P. G.; FITZGERALD, J.; WOODCOCK, J.; PELESKA, J. Systems of systems engineering: Basic concepts, model-based techniques, and research directions. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 48, n. 2, Sep. 2015. ISSN 0360-0300. Available: <https://doi.org/10.1145/2794381>. Citation on page 31.

OLIVEIRA, A. L. d. **A model-based approach to support the systematic reuse and generation of safety artefacts in safety-critical software product line engineering**. Phd Thesis (PhD Thesis) — Universidade de São Paulo, 2016. Citation on page 39.

OLIVEIRA, A. L. de; BRAGA, R.; MASIERO, P.; PARKER, D.; PAPADOPOULOS, Y.; HABLI, I.; KELLY, T. Variability management in safety-critical systems design and dependability analysis. **Journal of Software: Evolution and Process**, v. 31, n. 8, p. e2202, 2019. E2202 smr.2202. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2202>. Citation on page 37.

OMG. Service oriented architecture modeling language (soaml) specification. **Object Management Group**, 2012. Citation on page 83.

OMG. **OMG Systems Modeling Language SysML**. 2017. Available: <https://www.omg.org/spec/SysML/1.6/PDF>. Citations on pages 67, 79, and 83.

____. **Unified Modeling Language (UML) 2.5**. 2017. Available: <https://www.omg.org/spec/UML/2.5.1/PDF>. Citations on pages 67, 79, and 83.

____. **Meta-Object Facility**. 2019. Available: <http://https//www.omg.org/mof/,last>. Citation on page 68.

PAIGE, R. F.; ROSE, L. M.; GE, X.; KOLOVOS, D. S.; BROOKE, P. J. Fptc: automated safety analysis for domain-specific languages. In: SPRINGER. **International Conference on Model Driven Engineering Languages and Systems**. [S.l.], 2008. p. 229–242. Citation on page 28.

PAPADOPOULOS, Y. **HIP-HOPS Manual**. [S.l.], 2013. Citation on page 44.

PAPADOPOULOS, Y.; MCDERMID, J. A. Hierarchically performed hazard origin and propagation studies. In: SPRINGER. **International Conference on Computer Safety, Reliability, and Security**. [S.l.], 1999. p. 139–152. Citations on pages 27 and 48.

PAPADOPOULOS, Y.; WALKER, M.; PARKER, D.; RÜDE, E.; HAMANN, R.; UHLIG, A.; GRÄTZ, U.; LIEN, R. Engineering failure analysis and design optimisation with hip-hops. **Engineering Failure Analysis**, v. 18, n. 2, p. 590–608, 2011. ISSN 1350-6307. Citations on pages 26, 27, 45, 46, 47, 48, 67, 75, and 79.

PINTO, C. A.; MCSHANE, M. K.; BOZKURT, I. System of systems perspective on risk: towards a unified concept. **International Journal System of Systems Engineering**, v. 3, n. 1, p. 33–46, 2012. Citations on pages 53, 57, 58, 59, and 60.

PMI (Ed.). **A Guide to the Project Management Body of Knowledge (PMBOK Guide)**. 6. ed. Newtown Square, PA: Project Management Institute, 2017. Citations on pages 17, 59, and 60.

PROCHAZKOVA, D. Identification and management of risks of system of systems. **International Journal of Computer an Information Technology**, Citeseer, p. 2279–0764, 2013. Citations on pages 58, 59, 63, and 65.

PUMFREY, D. J. **The principled design of computer system safety analyses**. Phd Thesis (PhD Thesis) — University of York, 1999. Citations on pages 43 and 45.

RABINER, L.; JUANG, B. An introduction to hidden markov models. **IEEE ASSP Magazine**, v. 3, n. 1, p. 4–16, 1986. Citation on page 75.

REDMOND, P. **A system of systems interface hazard analysis technique**. [S.l.], 2007. Citations on pages 28, 40, 48, 62, 64, 65, 67, and 73.

REDMOND, P. J.; MICHAEL, J. B.; SHEBALIN, P. V. Interface hazard analysis for system of systems. In: IEEE. **2008 IEEE International Conference on System of Systems Engineering**. [S.l.], 2008. p. 1–8. Citations on pages 67 and 107.

REICH, J.; SCHNEIDER, D. Towards (semi-) automated synthesis of runtime safety models: A safety-oriented design approach for service architectures of cooperative autonomous systems. In: SPRINGER. **International Conference on Computer Safety, Reliability, and Security**. [S.l.], 2018. p. 139–150. Citations on pages 92 and 93.

REICH, J.; SCHNEIDER, D.; ADLER, R.; WEI, R.; KELLY, T.; SOROKOS, I.; ZELLER, M.; GUO, J.; KAUKEWITSCH, C.; MACHER, G.; ARMENGAUD, E. Digital dependability identities and the open dependability exchange meta-model. **DEIS Project**, 2020. Citation on page 91.

SABERI, A. K.; BARBIER, E.; BENDERS, F.; BRAND, M. V. D. On functional safety methods: A system of systems approach. In: IEEE. **2018 Annual IEEE International Systems Conference (SysCon)**. [S.l.], 2018. p. 1–6. Citation on page 28.

SAE. **Architecture Analysis and Design Language (AADL)**. [S.l.], 2017. Available: <https://www.sae.org/standards/content/as5506c/>. Citations on pages 67 and 79.

SAE, A. 4761: Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment. **Society of Automotive Engineers, Inc**, 1996. Citation on page 39.

SALADO, A. Abandonment: A natural consequence of autonomy and belonging in systems-of-systems. In: IEEE. **2015 10th System of Systems Engineering Conference (SoSE)**. [S.l.], 2015. p. 352–357. Citations on pages 58 and 61.

____. Exile: A natural consequence of autonomy and belonging in systems-of-systems. In: IEEE. **2016 Annual IEEE Systems Conference (SysCon)**. [S.l.], 2016. p. 1–5. Citations on pages 58 and 61.

SEVCIK, F. Current and future concepts in fmea (failure modes and effects analysis). In: **Annual Reliability and Maintainability Symposium, Philadelphia, Pa**. [S.l.: s.n.], 1981. p. 414–421. Citation on page 43.

SHABAN, A.; ABDELWAHED, A.; Di Gravio, G.; AFEFY, I. H.; PATRIARCA, R. A systems-theoretic hazard analysis for safety-critical medical gas pipeline and oxygen supply systems. **Journal of Loss Prevention in the Process Industries**, v. 77, p. 104782, 2022. ISSN 0950-4230. Available: <https://www.sciencedirect.com/science/article/pii/S0950423022000596>. Citation on page 25.

SHAH, P.; DAVENDRALINGAM, N.; DELAURENTIS, D. A. A conditional value-at-risk approach to risk management in system-of-systems architectures. In: IEEE. **2015 10th System of Systems Engineering Conference (SoSE)**. [S.l.], 2015. p. 457–462. Citations on pages 57, 58, 59, and 65.

SHARVIA, S.; KABIR, S.; WALKER, M.; PAPADOPOULOS, Y. Chapter 12 - model-based dependability analysis: State-of-the-art, challenges, and future outlook. In: MISTRIK, I.; SOLEY, R.; ALI, N.; GRUNDY, J.; TEKINERDOGAN, B. (Ed.). **Software Quality Assurance**. Boston: Morgan Kaufmann, 2016. p. 251–278. ISBN 978-0-12-802301-3. Available: <https://www.sciencedirect.com/science/article/pii/B9780128023013000120>. Citation on page 26.

SIMPLEMAN, L.; MCMAHON, P.; BAHNMAIER, B.; EVANS, K.; LLOYD, J. **Risk management guide for DOD acquisition**. [S.l.], 1998. Citations on pages 62 and 65.

SOMMERVILLE, I.; DEWSBURY, G.; CLARKE, K.; ROUNCEFIELD, M. Dependability and trust in organisational and domestic computer systems. In: ____. **Trust in Technology: A Socio-Technical Perspective**. [S.l.]: Springer Netherlands, 2006. p. 169–193. Citations on pages 61 and 62.

THAPALIYA, A.; KWON, G. A unified approach for uml based safety oriented level crossing using fta and model checking. In: **Proceedings of the 19th Korea Conference on Software Engineering (KCSE 2017)**. [S.l.: s.n.], 2017. v. 19, p. 89–90. Citation on page 43.

VESELY, W.; DUGAN, J.; FRAGOLA, J.; MINARICK, J.; RAILSBACK, J. Fault tree handbook with aerospace applications. **NASA Office of Safety and Mission Assurance**, 2002. Citation on page 42.

WALLACE, M. Modular architectural representation and analysis of fault propagation and transformation. **Electronic Notes in Theoretical Computer Science**, v. 141, n. 3, p. 53 – 71, 2005. ISSN 1571-0661. Proceedings of the Second International Workshop on Formal Foundations of Embedded Software and Component-based Software Architectures (FESCA 2005). Available: <http://www.sciencedirect.com/science/article/pii/S1571066105051650>. Citations on pages 46 and 47.

WEI, R.; KELLY, T. P.; HAWKINS, R.; ARMENGAUD, E. Deis: Dependability engineering innovation for cyber-physical systems. In: SPRINGER. **Federation of International Conferences on software technologies: applications and foundations**. [S.l.], 2017. p. 409–416. Citation on page 27.

WINTHER, R.; JOHNSEN, O.-A.; GRAN, B. A. Security assessments of safety critical systems using hazops. In: SPRINGER. **International Conference on Computer Safety, Reliability, and Security**. [S.l.], 2001. p. 14–24. Citation on page 41.

WOJCIK, L.; HOFFMAN, K. Systems of systems engineering in the enterprise context: a unifying framework for dynamics. In: **System of Systems Engineering, 2006 IEEE/SMC International Conference on**. [S.l.: s.n.], 2006. p. 8 pp.–. Citations on pages 25 and 31.

APPENDIX

A

TRAFFIC LIGHT ASSISTANT LOCAL FAILURE DATA

In this appendix, CS and component local failure data are defined. Table 15 and Table 16 show the output deviation for each CS output port, whereas Table 17 and Table 18 show the output deviation for each component output port.

Table 15 – CS Local Failure Data – Part 1.

System	**Output Deviation**	**Causes**
Elec. Drive System	Omission- ElecDriveSystem. acc	Omission- GasPedal.acc OR InternalFailure
ADAS Basis System	Omission- ADASBasisSystem. gas	Omission- EVAccelerationController. gas OR InternalFailure
ADAS Basis System	Omission- ADASBasisSystem. brake	Omission- EVAccelerationController. brake OR InternalFailure
ADAS Basis System	Omission- ADASBasisSystem. obstacles2	Omission- EnvPerceptionSystem. obstacles2 OR InternalFailure
Env. Perception System	Omission- EnvPerceptionSystem. obstacles2	Omission- Radar.obstacles2 OR InternalFailure

Source: Elaborated by the author.

Table 16 – CS Local Failure Data – Part 2.

System	**Output Deviation**	**Causes**
Traffic Light Assistant System	Omission-TrafficLightAssistantSystem.optimalAcc	Omission-EVSpeedCalculator.optimalAcc OR InternalFailure
Front Vehicle System	Omission-FrontVehicleSystem.obstacles1	Omission-Sensor.obstacles1 OR InternalFailure
Intersection Control System	Omission-IntersectionControlSystem.timeUNRedTL	InternalFailure
Intersection Control System	Omission-IntersectionControlSystem.durationNRedTL	InternalFailure
Intersection Control System	Omission-IntersectionControlSystem.posNextTL	InternalFailure

Source: Elaborated by the author.

Table 17 – Component Local Failure Data – Part 1.

Component	**Output Deviation**	**Causes**
Gas Pedal	Omission-GasPedal.acc	Omission-GasPedal.gas OR InternalFailure
EV Acceleration Controller	Omission-EVAccelerationController.gas	Omission-EVAccelerationController.optimalAcc OR InternalFailure
EV Acceleration Controller	Omission-EVAccelerationController.brake	Omission-EVAccelerationController.optimalAcc OR InternalFailure
Radar	Omission-Radar.obstacles2	Omission-Radar.env OR InternalFailure
EV Speed Calculator	Omission-EVSpeedCalculator.optimalAcc	Omission-EVSpeedCalculator.evEOSpeed OR Omission-EVSpeedCalculator.evSafeSpeed OR InternalFailure

Source: Elaborated by the author.

Table 18 – Component Local Failure Data – Part 2.

Component	**Output Deviation**	**Causes**
Safe EV Speed Profile Calculator	Omission-SafeEVSpeedProfileCalculator.evSafeSpeed	Omission-SafeEVSpeedProfileCalculator.obstacles1 OR Omission-SafeEVSpeedProfileCalculator.evState OR Omission-SafeEVSpeedProfileCalculator.obstacles2 OR InternalFailure
EV Localization Sensor	Omission-EVLocalizationSensor.evState	InternalFailure
Sensor	Omission-Sensor.obstacles1	Omission-Sensor.env OR InternalFailure
EO-EV Speed Profile to TL Calculator	Omission-EOEVSpeedProfileToTLCalculator.evEOSpeed	Omission-EOEVSpeedProfileToTLCalculator.timeUNRedTL OR Omission-EOEVSpeedProfileToTLCalculator.durationNRedTL OR Omission-EOEVSpeedProfileToTLCalculator.posNextTL OR Omission-EOEVSpeedProfileToTLCalculator.evState OR InternalFailure

Source: Elaborated by the author.


ICMC
USP
SÃO CARLOS