Quality evaluation model for crisis and emergency management Systems-of-Systems

Daniel Soares Santos

Daniel Soares Santos

Quality evaluation model for crisis and emergency management Systems-of-Systems

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. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Elisa Yumi Nakagawa

USP - São Carlos
January 2017

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

```
        Santos, Daniel Soares
S237q      Quality evaluation model for crisis and
        emergency management Systems-of-Systems / Daniel
        Soares Santos; orientadora Elisa Yumi Nakagawa. --
        São Carlos, 2017.
           118 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, 2017.


           1. Software Quality . 2. Evaluation Model. 3.
        Systems-of-Systems. I. Nakagawa, Elisa Yumi,
        orient. II. Título.
```

Daniel Soares Santos

Modelo de avaliação de qualidade para Sistemas-de-Sistemas de gerenciamento de crises e emergências

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

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Elisa Yumi Nakagawa

USP - São Carlos
Janeiro de 2017

Abstract

Systems-of-Systems (SoS) have performed an important and even essential role to the whole society and refer to complex software-intensive systems, resulted from interoperability of independent constituent systems that work together to achieve more complex missions. SoS have emerged specially in critical application domains and, therefore, high level of quality must be assured during their development and evolution. However, dealing with quality of SoS still presents great challenges, as SoS present a set of unique characteristics that can directly affect the quality of such systems. Moreover, there are not comprehensive models that can support the quality evaluation of SoS. Motivated by this scenario, the main contribution of this Master's project is to present a SoS Evaluation Model, more specifically, addressing the crisis/emergency management domain, built in the context of a large international research project. The proposed model covers important evaluation activities and considers all SoS characteristics and challenges not usually addressed by other models. This model was applied to evaluate a crisis/emergency management SoS and our results have shown it viability to the effective management of the SoS quality.

Keywords: Software Quality, Evaluation Model, Crisis and Emergency Management, Systems-of-Systems.

Resumo

Sistemas-de-Sistemas (SoS, do inglês Systems-of-Systems) realizam um importante e até essencial papel na sociedade. Referem-se a complexos sistemas intensivos em software, resultado da interoperabilidade de sistemas constituintes independentes que trabalham juntos para realizar missões mais complexas. SoS têm emergido especialmente em domínios de aplicação crítica, portanto, um alto nível de qualidade deve ser garantido durante seu desenvolvimento e evolução. Entretanto, lidar com qualidade em SoS ainda apresenta grandes desafios, uma vez que possuem um conjunto de características únicas que podem diretamente afetar a qualidade desses sistemas. Além disso, não existem modelos abrangentes para o suporte à avaliação de qualidade de SoS. Motivado por este cenário, a principal contribuição deste projeto de mestrado é apresentar um modelo de avaliação para SoS, especialmente destinado ao domínio de gerenciamento de crises e emergências. Este modelo foi construído no contexto de um grande projeto de pesquisa internacional, e cobre as mais importantes atividades de avaliação, considerando as principais características e desafios de SoS geralmente não abordados por outros modelos. Este modelo foi aplicado na avaliação de um SoS de gerenciamento de crises e emergência, e nossos resultados têm mostrado sua viabilidade para o efetivo gerenciamento da qualidade de SoS.

Palavras-chave: Qualidade de Software, Modelo de Avaliação, Gerenciamento de Crises e Emergências, Sistemas-de-Sistemas.

Contents

List of Figures

List of Tables

Chapter

1

Introduction

Software-intensive systems have become increasing ubiquitous and complex over the last years and have evolved to enable the cooperation of various operationally independent and distributed systems, which, in many cases, are developed with different technologies and for a series of platforms. Cooperation is a major concern that enables systems to solve larger, complex missions, which could not be solved by any system working alone (Maier, 1998). Referred as Systems-of-systems (SoS), this class of system emerges as a result of the interaction of large, complex, and independent systems (called constituent systems in the context of SoS) (Maier, 1998). This class of systems has become a trend of the current software-intensive systems and is grounding the path for future systems (Nakagawa et al., 2013). They can be found in many sectors and application domains, mainly critical ones. Good examples of SoS can be found in representative domains, such as robotics, airport, avionics, smart-* systems (e.g., smart-grid, smart-cities, smart-buildings, and smart-farms), and in the military domain (e.g., aerospace, warfare, and global earth observation) (Bianchi et al., 2015; Jamshidi, 2008; Lane, 2013; Nakagawa et al., 2013). Differently from classical software systems, SoS present a set of characteristics that make them unique (Maier, 1998): (i) operational independence of constituent systems; (ii) managerial independence of constituent systems; (iii) evolutionary development; (iv) emergent behavior from the collaboration of constituents; and (v) distribution of the constituent systems, geographically or not. These SoS characteristics have been widely shared among the international community and help to understand the difference between SoS and complex monolithic systems.

SoS can also be classified accordingly to the level of interaction among their constituents (Maier, 1998). Directive SoS have a central controller, a common mission, and dependence among constituents, while in another extreme, virtual SoS have no central controller and a mission emerges among constituents that do not have a strict dependence among themselves. Each type of SoS requires specific solutions, which considerably impose challenges for their development (Maier, 1998).

Specifically for the context of SoS, mainly because of their applicability in several critical application domains, there is a great necessity of dealing with software quality during their development and evolution (DoD, 2010). The quality of any software systems is directed related to their capacity of satisfying stated goals and needs that justify their development (Wagner, 2013). In addition to perform a set of required functions, systems must exhibit some quality characteristics (generally called quality attributes) related to the needs of stakeholders (people or organizations)(Wagner, 2013). Normally, the SoS application domains are responsible for presenting a required set of these attributes, which can be different from the quality attributes considered for developing constituent systems. For example, the avionics domain requires that systems exhibit a high level of dependability among all systems communication to decrease chances of issues in airplanes during flying, whereas the main concern of navigation systems could be the reliability for defining the position of airplanes. Despite the many ways to increase the quality of SoS throughout their life cycle, it is important to evaluate their quality to ensure they will be able to achieve their mission. Otherwise, more iterations should be carried until enough quality is achieved for deployment.

One way to evaluate the quality of systems is to use a set of standards established by international organizations, such as ISO[1] (International Organization for Standardization), IEC[2] (International Electrotechnical Commission), and IEEE[3] (Institute of Electrical and Electronics Engineers). When used in combination, standards allow to assess the quality of software systems in a systematic way. ISO/IEC 25010 (2011) provides two quality models composed of quality attributes of software systems as a product (e.g., performance and maintainability) and under operation by users (e.g., safety and satisfaction). These attributes serve as basis for specifying quality requirements and quality evaluation(Wagner, 2013). Besides that, ISO/IEC 9126-2 (2003), ISO/IEC 9126-3 (2003), and ISO/IEC 9126-4 (2003) provide a set of metrics to measure internal, external, and quality in use attributes, respectively, such as the presented in the quality models of ISO/IEC 25010. Finally, ISO/IEC 25040 (2011) describes a process (and its requirements) for evaluating the quality of software systems. It is composed of five phases and can be supported

[1]`http://www.iso.org`
[2]`http://www.iec.ch`
[3]`https://www.ieee.org`

by the standards ISO/IEC 9126-2, ISO/IEC 9126-3, and ISO/IEC 25010 throughout its application.

1.1 Motivation

The main problem that is found and has motivated the conduction of this Master's project is that the quality management of SoS still presents great challenges for the classical software engineering approaches, as SoS present a set of unique characteristics (DoD, 2010; Larsson et al., 2016; Wang and Wan, 2014). For SoS, besides interoperability, several other quality attributes are critical (e.g., performance, reliability, and security) and, therefore, should be carefully considered so that the objectives of the SoS can be reached (Ackermann et al., 2009). However, achievement of these goals is quite difficult, as the constituents are sometimes developed and maintained by different organizations, with their own stakeholders, development teams, processes, and resources (Gagliardi et al., 2009; Santos et al., 2014). In addition, because the size, complexity and common asynchronous development of SoS becomes difficult to perform evaluation activities in this environment (DoD, 2008; Wang and Wan, 2014).

Currently, there are no comprehensive evaluation models for SoS that can control and improve the quality of such systems. In addition, there are no specific studies or wide experience reports to guide the quality evaluation of an SoS, considering all of its characteristics and, mainly, the challenges inherent of this context. Most studies about quality evaluation of SoS address the architecture level, which does not eliminate the need of a product quality evaluation (Santos et al., 2014). As previously mentioned, a common way to evaluate the quality of a software product is to use general purpose quality models or evaluation processes (e.g., ISO/IEC 25010/25040). However, their application is limited in the context of SoS, where it is needed to consider the evaluation not only in system product level but also in SoS level, besides consider the impact of each independent constituent in the achievement of the SoS mission. In addition, these standards do not discuss in an sufficient way how the evaluation results could be aggregated to obtain the whole quality of a system, much less a SoS.

1.2 Objective

Motivated by this scenario, the main contribution of this Master's project is to propose a quality evaluation model suitable to SoS, addressing the crisis/emergency management domain. This quality evaluation model was built in the context of a large international research project entitled RESCUER project[4]. The proposed model intends to cover important evaluation activities, considering all SoS characteristics and main challenges related to their evaluation.

The proposed evaluation model restructures the common evaluation activities considering both constituent system and SoS levels, providing guidelines to conduct evaluation activities in light of SoS characteristics. To support the use of our evaluation model, we established a quality model extending the current ISO/IEC 25010 with a way to link constituent systems with the quality attributes and their priority concerning the achievement of SoS mission. In addition, we tailored the ISO/IEC 25010 and ISO/IEC 9126-2, ISO/IEC 9126-3, and ISO/IEC 9126-4 to crisis/emergency management domain through the contribution of RESCUER stakeholders, defining a specific quality model and metrics for that domain. In addition, we adapted the aggregation model proposed by Wagner et al. (2012) to SoS context to guide the aggregation of evaluation results of each constituent system to obtain the measurement of whole SoS quality. In summary, all artifacts produced during the evaluation phases proposed by this evaluation model (quality model, metrics, strategies, and plan) can be used as basis for evaluationg any crisis/emergency management SoS (CEMSoS).

Finally, we applied this model to evaluate a CEMSoS. Results achieved show the viability of our evaluation model, what has motivated us to apply this model in other SoS being developed in the context of our research group.

1.3 Organization

The remainder of this project is organized as follows. Chapter 2 presents the main concepts regarding the SoS and software quality evaluation. In addition, it presents an study about the commonly considered quality attributes in the SoS context. Chapter 3 details the SoS Evaluation Model proposed in this Master's project. Chapter 4 presents a case study applying the SoS Evaluation Model in a crisis and emergency management SoS. Finally, Chapter 5 describes our main conclusion, limitation, and future works.

[4]`http://www.rescuer-project.org`

Chapter

2

Background

This chapter aims to present the main concepts related to Systems-of-Systems (SoS) and Software Quality Evaluation. The goal is to provide the needed information to a complete understanding of the SoS Quality Evaluation Model proposed in this Master's project.

In addition, this chapter presents the main results of a Systematic Literature Review (SLR) - a special kind of literature review conducted to identify, assess, and interpret available evidences on a research question, topic or phenomenon of interest (Kitchenham and Charters, 2007) - about the commonly considered quality attributes in the context of SoS. Moreover, it provides an discussion about the five main quality attributes and the challenges involved in their evaluation. Main results of this SLR are presented in Section 2.3, whereas Appendix A presents its protocol and conduction process. Results of this SLR were published as a paper in the 3th International Workshop on Software Engineering and Systems-of-Systems (Bianchi et al., 2015) and as a technical report in University of São Paulo (Santos et al., 2015a).

Due to difficulty to perform tests in an environment of asynchronous development (DoD, 2008) such as the commonly found in the SoS context, most of the SoS evaluation experiences reported in literature are about architecture evaluation of these systems. It happens due to not only the size and complexity of SoS, but also the fact that parts of the functionality are deployed in different constituents (DoD, 2008). Despite the architecture evaluation field is not completely related to this Master's project theme, we also conducted an SLR to understand how this kind of evaluation is performed in the SoS context, and which quality attributes are considered. Results of this SLR are also interesting, since most SoS architecture evaluations can be conducted instead of SoS product evaluation due the its complexity. The protocol, conduction process, and main results of this SLR are presented in Appendix B. Results of this SLR was published in the 8th Workshop on Distributed Software Development, Software Ecosystems, and Systems-of-Systems (Santos et al., 2014).

This chapter is organized as follows. Section 2.1 presents the main concepts about Systems-of- Systems domain. Section 2.2 describes the main concepts about the software quality evaluation area, with an special focus on quality models, metrics, evaluation techniques and process. Section 2.3 describes a study about the commonly considered quality attributes in the SoS context. Finally, Section 2.4 presents the final remarks on the topics covered in this chapter.

2.1 Systems-of-Systems

A System-of-Systems (SoS) is result of the interaction of systems that operate independently but collaborate among them to achieve goals that would not be possible by these systems working separately (DoD, 2008; Luzeaux and Ruault, 2010; Maier, 1998). This scenario is increasingly common and such systems are becoming important and even essential to the whole society. However, development and quality management of SoS are still a current challenge that has not still been overcome by the classical software engineering (DoD, 2010; Larsson et al., 2016; Wang and Wan, 2014).

The first definitions to the term SoS appeared in studies published in the early 90s in the areas of defense, information technology, and air traffic control (Luzeaux and Ruault, 2010). Although the term meets featured today, it has no universally recognized definition (Luzeaux and Ruault, 2010), and then, their characterization depends often on the viewpoint and system's context. A practical definition is that an SoS is a "super-system" composed of complex and operationally independent systems to achieve higher goal (Jamshidi, 2008). Due to the increasing spread of the concepts of SoS in software engineering (Boehm and Lane, 2006), several other SoS definitions can be found in the literature, such as:

- Maier (1998): *"A system of systems is an assemblage of components which individually may be regarded as systems, and which possess two additional properties: operational independence of the components and managerial independence of the components."*

- ISO/IEC 24765 (2010): *"A large system that delivers unique capabilities, formed by integrating independently useful systems."*

- Sage and Cuppan (2001): *"Systems of systems exist when there is a presence of a majority of the following five characteristics: operational and managerial independence, geographic distribution, emergent behavior, and evolutionary development."*
- Eisner (1993): *"Systems of systems are large geographically distributed assemblages developed using centrally directed development efforts in which the component systems and their integration are deliberately, and centrally, planned for a particular purpose."*
- Minkiewicz (2006): *"The constituents of a system of systems are independently useful but synergistically superior when acting in concert. In other words, it represents a collection of systems whose capabilities, when acting together, are greater than the sum of the capabilities of each system acting alone."*

As previously mentioned, there is still no universally recognized definition for SoS. However, it can be perceived that most of definitions converge to a set of key characteristics previously identified by Maier (1998).

2.1.1 SoS Characteristics

The following characteristics have been widely shared among the international community and help to understand the difference between SoS and complex monolithic systems (Maier, 1998):

- **Operational Independent:** constituent systems are operationally independents and have their own goals, even when disconnected from the SoS;
- **Managerial Independent:** managerial independence means that each constituent system is developed and maintained by different organizations, with their own stakeholders, development teams, processes, and resources;
- **Evolutionary and Adaptive Development:** SoS development is evolutionary and adaptive, where structures, functions, and purposes are added, removed, and modified according to emergence needs of the system;
- **Emergent Behavior:** emergent behavior means that properties, functionality, and features of the SoS can not be identified or provided by any constituent system working separately; and
- **Geographical distribution:** constituent systems may be located in different places depending on the employed technologies and communication means.

Despite the dissemination of characteristics of SoS in the community, the establishment of the exact differences between monolithic systems and SoS is a more delicate topic than it seems. Luzeaux and Ruault (2010) present the following reflection about this issue: A computer is a system composed of subsystems (peripheral devices, central unit), where each subsystems are independently developed by different organizations, and some of then can be put to different use, however, a computer is not generally recognized as SoS. On the other hand, everyone agrees the air transportation system is an SoS, but it can also be classified as a system? Opinions differ depending on the system concept accepted by each one.

According to Oberndorf and Sledge (2010), the major difference between systems and SoS is regarding their span of control. For a monolithic system, practices and processes are determined within a single program. For SoS, instead, these practices and processes are influenced by the SoS as a whole. To better understand this difference, Oberndorf and Sledge (2010) compares some specific characteristics of monolithic systems and SoS regarding some perspectives (see Table 2.1).

Table 2.1: SoS vs. monolithic systems (Oberndorf and Sledge, 2010)

Comparison Perspective	**Monolithic Systems**	**System-of-Systems**
Problems	Causes of problems and behaviors effects can be largely known	Problems are affected by a combination of factors, some are known, some unknown, and some unknowable.
Dependencies	System dependencies are mainly within one program context and known scope control. Changes that affect dependencies within a system can be controlled	SoS dependencies are outside the scope of control and the context may not be completely known. SoS changing dependencies may be only managed.
System focus	The focus is to optimize their capabilities	The focus is achieving the collective emergent capabilities.
Negotiations and decisions	Negotiations and decisions are performed within the program purview	The boundaries are not clear. There is more dependence on collaboration and influence in negotiations and decisions.

In general, when compared to single systems, an SoS requires new paradigms due to the independent management, operation, development, and evolution of the constituent systems, and mainly because their focus on collective emergent capabilities (Oberndorf and Sledge, 2010).

SoS started to gain popularity mainly on military systems as a strategy to reach goals or deliver unique capabilities wherein a collaborative work of complex systems is needed (DoD, 2010; Maier, 1998). Currently, SoS have increasingly become the focus of interest in Software Engineering (Boehm and Lane, 2006) and, consequently, new application scenarios have emerged in research and industry.

2.1.2 Types of System-of-Systems and Some Examples

SoS have been commonly applied in safety-critical domains in which failures may cause death or injury to people, harm to the environment, or substantial economic loss (Bozzano and Villafiorita, 2010). Besides the military domain, some examples are (Lane, 2013): Car2- Car interactions that help to prevent accidents at crossings or optimize cruise speed; Ambient Assisted Living (AAL) solutions that assist people with specific needs at home and while travelling via monitoring, remote control or automation; and emergency and disaster management systems that allow more efficient response to crisis and incidents through integration of police, firefighters, military, and medical systems.

Despite the most of system are developed without explicit concerns about SoS consideration (DoD, 2008), these systems can be classified based upon the level of responsibility and authority overseeing the evolution of the SoS (Lane, 2013). In this sense, Maier (1998) and Dahmann and Baldwin (2008) identified four types of SoS: virtual, collaborative, acknowledged, and directed.

- **Virtual:** Constituent systems are not subjected to a central management authority and there is not a clear SoS purpose. Constituent systems are not necessarily known and mechanisms to maintain the SoS are not evident. An example of this type of SoS is the World Wide Web and all provided services that are integrated in an ad hoc manner;

- **Collaborative:** Constituent systems interact somehow voluntarily to reach shared or common purposes. There is not an authority to guide or manage activities of the constituent systems. Examples of this are the emergency and disaster management systems, where each agency voluntarily cooperates to allow a more efficient response to crisis, and incidents;

- **Acknowledged:** Despite the considerable independence of the constituent systems, there are specific objectives, management team, and resources to address concerns on the SoS level. Further changes in the constituent systems are based on their collaboration within the SoS. Examples of this type of SoS are military command and control systems. According to Lane (2013), these systems have transitioned from collaborative to acknowledged SoS due to the importance of the missions and complexity of the cross-cutting SoS capabilities; and

- **Directed:** Directed SoS are centrally built and managed to fulfill specific purposes. Although constituent systems maintain the operational independence, their behavior and evolution are predominantly subordinated to a central purpose. A healthcare SoS is an example of this type of SoS. These systems integrate several patient care systems, such as patient management system, imaging management system, pharmacy system, among others.

As SoS have been commonly applied in safety-critical domains, their quality need to be carefully evaluated to ensure that SoS mission be sufficiently achieved. Next section presents main concepts related to software quality evaluation.

2.2 Software Quality Evaluation

Software quality has been a research topic widely investigated over the last three decades (Wagner, 2013). Due this, concepts and definitions about this theme has been improved and evolved over the years. Currently, quality is accept as a complex and multifaceted concept that can be described in different perspectives (Wagner, 2013). Garvin (1984), for example, defines a set of different views or approaches to quality, which also can be used to understand the software quality:

- The transcendental approach sees quality as something that can be recognized but not defined. It captures the intuitive feeling of quality that can be useful in software product quality management, but it cannot be objectively measured;
- The user approach sees quality as fitness for purpose. It assumes that a product with quality needs to satisfy the user requirements and mainly the users expectation;
- The value-based approach sees quality as dependent on the value a customer is willing to pay for it. In this approach, the conformance and non conformance to requirements are associated to the benefits and cost to do it. A cheap but less durable product can be of higher value to a user than a durable but very expensive product;
- The product approach sees quality as a inherent characteristics of the product. In this approach, the quality is achieved when a set of desired attributes is sufficiently identified in the product. Therefore, it is assumed that each attributes is known, describable, and precisely measurable; and
- The manufacturing approach sees quality as conformance to specification. It assumes the possibility to define the requirements of a product completely and, consequently, identification of any deviation. Because this, the development process needs to ensure that the software is created in a way that conforms to its specification.

It can be perceived that each one of these approaches can be more relevant and adequate at different period of the product's life cycle, since its inception and requirement specification until to product's building (Kitchenham and Pfleeger, 1996). This variety of quality perceptions also reflects the slightly different meanings of quality used currently in international software standards. For example, ISO/IEC 24765 (2010) defines quality with six definition:

- The degree to which a system, component or process meets specified requirements;
- The ability of a product, service, system, component or process to meet customer or user needs, expectations or requirements;
- The totality of characteristics of an entity that bears on its ability to satisfy stated and implied needs;
- Conformity to user expectations, conformity to user requirements, customer satisfaction, reliability, and level of defects present;
- The degree to which a set of inherent characteristics fulfils requirements; and
- The degree to which a system, component or process meets customer or user needs or expectations.

The most definitions address the idea of requirements that need to be satisfied to achieve quality in a system, component, process, product or service (Wagner, 2013). In order to apply this quality concept in the software's life cycle, basically, we need to understand the user expectations, translate them into clear product attributes and ensure that these attributes are then implemented in the product (Wagner, 2013). This process is generally defined as software quality control (ISO/IEC 24765, 2010). In this context, quality evaluation is an important part of quality control, where a systematic examination to verify if the product is capable of fulfilling specifics quality attributes is performed (ISO/IEC 24765, 2010). In order to measure how much the software product complies with a specific quality attribute, quality metrics must be defined. As result of the application of a quality metric to a software product, a qualitative value is obtained that characterizes the degree of compliance of the system to the quality attributes. More details about quality attributes and quality metrics are described in Section 2.2.1. A software quality model is basically used to support a better understanding and management of software quality through its hierarchical decomposition in quality attributes. Consequently, after the measurement of each expected quality attribute, the total quality of the system can be obtained through the its hierarchical composition.

2.2.1 Software Quality Models

A well-accepted way to support quality control is to adopt software quality models that have been over the last three decades an area widely researched in Software Engineering (AL-Badareen et al., 2011; Wagner, 2013). A quality model intends to describe, assess, and/or predict quality, usually through hierarchical decomposition of the general product quality into sub characteristics to make them better understandable and manageable (Wagner, 2013).

A software quality model may support a better understanding about what quality is in the context of software systems, supporting diverse activities throughout system development cycle. This is done through the identification of quality characteristics that are exhibited by software systems and can be aggregated to compose the overall software quality concept. These characteristics are generally called quality attributes (e.g., maintainability, performance, and security), which are presented by quality models to define, assess, and/or predict software quality (Wagner, 2013).

Quality models can be used by developers, acquirers, quality assurance, control staff, and independent evaluators, particularly those responsible for specifying and evaluating software product quality (ISO/IEC 25010, 2011). In summary, the use of quality models can benefit several software development activities, such as (ISO/IEC 25010, 2011):

- identifying software and system requirements;
- validating the comprehensiveness of a requirements definition;
- identifying software and system design objectives;
- identifying software and system testing objectives;
- identifying quality control criteria as part of quality assurance;
- identifying acceptance criteria for a software product and/or software-intensive computer system; and
- establishing measures of quality attributes to support these activities.

The first quality models emerged in the early days of the software engineering area and since then, quality models are still subject of research. Published in the 70s, Boehm et al. (1978) and McCall et al. (1977) proposed the first quality models for software. The two models are similar and use a hierarchical decomposition of quality in quality factors, such as maintainability and reliability (Wagner, 2013). The McCall's model was developed by the US Air-force Electronic System Decision (ESD), the Rome Air Development Center

(RADC), and General Electric (Ravichandran and Rothenberger, 2003). The major contribution of the McCall's model is the relationship between the quality attributes and metrics despite not consider directly the functionalities of software products. Boehm et al. (1978) added new quality factors to McCall's model (AL-Badareen et al., 2011), but no suggestion to improve the measuring of quality was presented. In 1987, Robert Grady and Hewlett-Packard Co. proposed the FURPS model (Grady and Caswell, 1987) that decomposed the software quality into functionality, usability, reliability, performance, and supportability. However, this model had considered only the user's quality aspects and not took into account quality attributes, such as portability and maintainability (AL-Badareen et al., 2011). Thereafter, several variations of these models have appeared over time to mitigate the shortcoming of the previous quality models. This motivated the world-wide standardization and consensus of the software quality model (AL-Badareen et al., 2011). In this context, ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) developed the ISO/IEC 9126 - a quality model that is part of the ISO/IEC 9000 standard, and is the most important standard for quality assurance. In 2011, the ISO/IEC 9126 was technically revised and replaced by ISO/IEC 25010 - Systems and software Quality Requirements and Evaluation (SQuaRE): System and software quality models.

The ISO/IEC 25010 quality model has become the standard for evaluating the quality of modern computer-intensive systems and is also used to measure the quality of U.S. Departament of Defense (DoD) systems (Azizian et al., 2011). ISO/IEC 25010 is based on the fact that the software product quality can be specified and evaluated using a hierarchical structure of quality attributes (ISO/IEC 25010, 2011).

In this model, the software quality attributes were classified in a hierarchical structure of characteristics and sub characteristics divided in two quality models (i.e., quality in use model, and product quality model), each one considering different quality aspects.

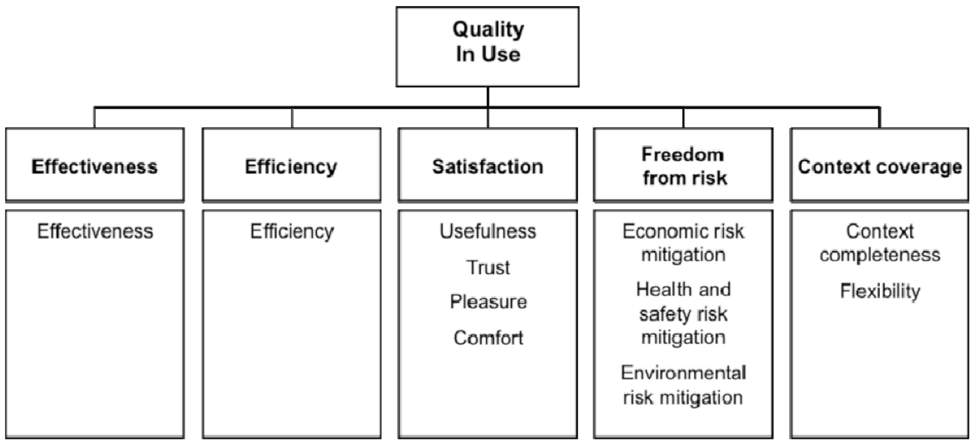


Figure 2.1: Quality in use model (ISO/IEC 25010, 2011)

The quality in use model defines five attributes related to outcomes of interaction with a system (ISO/IEC 25010, 2011): effectiveness, efficiency, satisfaction, freedom from risk, and context coverage, as showed in Figure 2.1. The quality in use of a system characterizes the impact that the system or software product has on stakeholders. It can be determined by the quality of the software, hardware and operating environment, and the characteristics of the users, tasks, and social environment (ISO/IEC 25010, 2011).

The product quality model categorizes system/software/product quality properties into eight attributes (ISO/IEC 25010, 2011): functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability. Each attribute is composed of a set of related sub attributes that are relevant and can be applied in both software and systems products (ISO/IEC 25010, 2011), as shown in Figure 2.2.

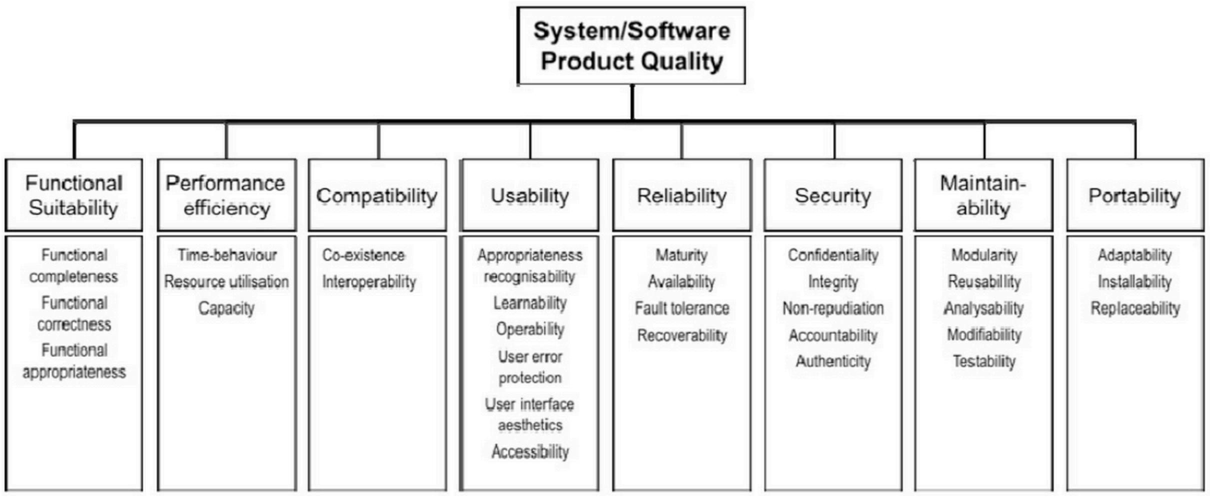


Figure 2.2: Product quality model (ISO/IEC 25010, 2011)

Both quality models are supposed to be applied to any kinds of computer systems that include a software product. In practical terms, when every sub attribute is measured, it is considered that the attribute is also measured by aggregation. Having every attributes measured, the overall quality of the product is determined. In order to achieve this goal, one or more metrics are defined and applied to each sub attribute, resulting in a value that represents the degree to which it is present in the final product.

2.2.2 Quality Metrics and Evaluation Techniques

When applying a quality metric, it is possible to obtain a quantitative value that characterizes the degree of compliance of the software to the corresponding quality attribute. The ISO/IEC 9126-2 - External Metrics (ISO/IEC 9126-2, 2003) and ISO/IEC 9126-4 - Quality In Use Metrics (ISO/IEC 9126-4, 2003) are examples of standard that present metrics for measuring sub attributes, and may be used together with ISO/IEC 25010 to evaluate quality of a software product. External metrics are used to measure the quality of the software product by measuring the system behavior, during testing stages or system operation (ISO/IEC 9126-2, 2003). On the order hand, quality in use metrics are applied in a realistic system environment to verify if a product meets the needs of specified users to achieve their goals (ISO/IEC 9126-4, 2003).

Table 2.2 presents examples of external metrics for two important quality sub attributes for any software system: (i) Functional completeness that refers the degree to which the set of functions covers all specified tasks and user objectives (ISO/IEC 25010, 2011); and (ii) Functional Appropriateness that refers the degree to which the functions facilitate the accomplishment of specified tasks and objectives (ISO/IEC 25010, 2011). In ISO/IEC 25010, these quality attributes compose the Functional Suitability attribute, which represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.

Table 2.2: Quality metric examples

Metric name	Purpose of the metrics	Method of application	Measurement	Interpretation
Functional adequacy metric	How adequate are the evaluated functions?	Number of functions that are suitable for performing the specified tasks comparing to the number of function evaluated.	X=1-A/B, A= Number of functions in which problems are detected in evaluation, B= Number of functions evaluated	0 <= X <= 1, The closer to 1.0, the more adequate.
Functional completeness metric	How complete is the implementation according to requirement specifications?	Do functional tests (black box test) of the system according to the requirement specifications. Count the number of missing functions detected in evaluation and compare with the number of function described in the requirement specifications.	X = 1 - A / B, A = Number of missing functions detected in evaluation, B = Number of functions described in requirement specifications	0<=X<=1 The closer to 1.0 is the better.

It can be perceived the application of both metrics will generate a quantitative result between 0 and 1 that will represent the degree of compliance with the evaluated quality attribute. Therefore, Functional Suitability quality attribute only will be achieved if the measurement of both quality sub attributes achieve acceptable results. These acceptable results (also referred as acceptance criteria) need to be defined in the system specification, since it will be used as reference to the system quality determination. In addition, it is very important to highlight that specific input data is needed for an adequate application of these metrics. Input data can be obtained by using different techniques, such as questionnaires, checklists, experiments, observations, software test, etc. For instance, for these metrics, ISO/IEC 9126-2 suggests to apply them during functional testing. Then, for detecting missing functions, it is suggested that each function stated in a requirement specification be tested one by one during functional testing. Such results become input to "Functional completeness metric". For detecting functions that are implemented but inadequate, it is suggested that each function be tested for multiple specified tasks. Such results become input to the "Functional adequacy metric".

The main idea is that the choice of the method or techniques that will be used to obtain the input data depends on the software product, evaluation requirements, evaluation purpose, importance and relevance of the quality attribute, etc (ISO/IEC 25040, 2011). In particular, the degree of rigour associated to the evaluation may be used as a guide to select evaluation techniques and it can be different to each software component and each quality attribute. As a consequence, evaluation at different levels gives different level of confidence in the quality of the software product (ISO/IEC 25040, 2011). For instance, ISO/IEC 25040 (2011) proposes, for each quality attribute, a list of evaluation techniques ranked from less demanding levels to more demanding levels, such as presented bellow:

- **Functionality:** 1. functional or black box testing; 2. inspection of development documentation guided by checklists; 3. unit testing with test coverage criteria.
- **Reliability:** 1. verification of the use of specific programming language facilities; 2. analysis of fault tolerance construct in the software design and code; 3. reliability growth modeling.
- **Usability:** 1. user interface and documentation inspection; 2. verification of the conformity to interface standards; 3. performing usage experiments with real users.
- **Efficiency:** 1. execution time measurement; 2. benchmark testing; 3. analysis of the design to determine the algorithmic complexity.
- **Maintainability:** 1. inspection of development documentation guided by checklists; 2. code measures and programming rules verification; 3. analysis of traceability between elements of development documentation.
- **Portability:** 1. analysis of software installation procedures; 2. programming rules verification; 3. analysis of software design.

As early mentioned, quality model is already a well-accepted way to support quality evaluation. However, for reliable results in quality evaluations, its planning, execution, and conclusion need to be performed in coherent and adequate way. In this sense, ISO/IEC 25040 defines a evaluation process as well as a set of roles to guide the software quality evaluation in different application contexts. In this evaluation process, all elements mentioned in this section are addressed in a synchronized way.

2.2.3 Quality Evaluation Process

Evaluation process is the basis for software product quality evaluation, since it defines a set of activities and roles needed to perform quality evaluation. The ISO/IEC 25040 -

Systems and software Quality Requirements and Evaluation (SQuaRE): Evaluation process - defines a software product quality evaluation reference model that includes the evaluation process as well as roles, such acquirer, developer organisation, or independent evaluator. This evaluation process has been widely accepted, since it can be used for different purposes and approaches as well as applied to evaluate the quality of pre-developed software or custom software during its development process (ISO/IEC 25040, 2011). Basically, ISO/IEC 25040 divides the evaluation process in five main activities: (i) Establish the evaluation requirements; (ii) Specify the evaluation; (iii) Design the evaluation; (iv) Execute the evaluation; and (v) Conclude the evaluation. As showed in Figure 2.3, each activity is divided in several tasks. Details about each activity and its tasks are provided following (ISO/IEC 25040, 2011):

Activity 1 - Establish the evaluation requirements: In this activity, requirements of the evaluation are established.

- Task 1.1 - Establish the purpose of the evaluation: Document the purpose for which the involved organization intend to evaluate the quality of the software product.
- Task 1.2 - Obtain the software product quality requirements: Identify stakeholders of the software product and specify the software product quality requirements using a quality model.
- Task 1.3 - Identify product parts to be included in the evaluation: Identify and document all product parts to be evaluated. The detail level and type of information (e.g., requirements specification, design diagrams, and test documentation) depends on the stage in the life cycle and purpose of the evaluation.
- Task 1.4 - Define the stringency of the evaluation: Define the evaluation rigor of the software product quality according to its intended use and purpose of the evaluation. The evaluation rigor will influence the evaluation techniques to be applied and evaluation results to be achieved.

Activity 2 - Specify the evaluation: In this activity, decision criteria for quality metrics are specified.

- Task 2.1 - Select quality measures: Select or define quality metrics to cover all software quality requirements. Measurement procedures should measure the software quality attributes (or sub attributes) with sufficient accuracy to allow its comparison with the acceptance criteria.

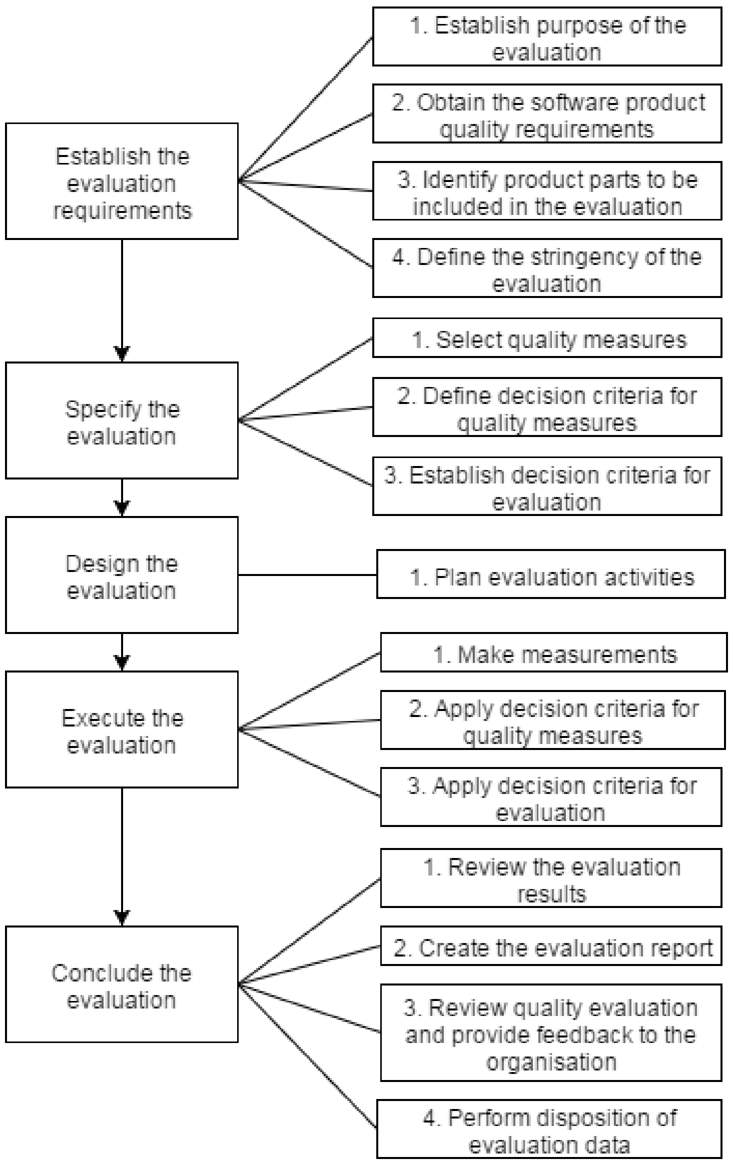

Establish the evaluation requirements
1. Establish purpose of the evaluation
2. Obtain the software product quality requirements
3. Identify product parts to be included in the evaluation
4. Define the stringency of the evaluation
Specify the evaluation
1. Select quality measures
2. Define decision criteria for quality measures
3. Establish decision criteria for evaluation
Design the evaluation
1. Plan evaluation activities
Execute the evaluation
1. Make measurements
2. Apply decision criteria for quality measures
3. Apply decision criteria for evaluation
Conclude the evaluation
1. Review the evaluation results
2. Create the evaluation report
3. Review quality evaluation and provide feedback to the organisation
4. Perform disposition of evaluation data

Figure 2.3: Evaluation process (ISO/IEC 25040, 2011)

- Task 2.2 - Define decision criteria for quality measures: Decision criteria or acceptance criteria shall be defined for the selected individual measures and are used to decide whether metric results are satisfactory or not, considering the expected results. These criteria are numerical thresholds or targets used to determine the need for action or further investigation.

- Task 2.3 - Define decision criteria for evaluation: the evaluator should prepare a procedure for further summarization, with separate criteria for different quality attributes in terms of their sub attributes and quality measures.

Activity 3 - Design the evaluation: In this activity, the evaluation plan is defined.

- Task 3.1 - Plan evaluation activities: The evaluation activities shall be scheduled taking into account the availability of resources and evaluation environment, such as personnel, software tools, computers, budget, evaluation methods, adopted standards, etc.

Activity 4 - Execute the evaluation: In this activity, the evaluation is executed, applying the quality metrics and decision criteria.

- Task 4.1 - Make measurements: Selected quality metrics shall be applied to the software product and components, according to the evaluation plan.
- Task 4.2 - Apply decision criteria for quality measures: Quality metrics results shall be compared with their defined decision criteria.
- Task 4.3 - Apply decision criteria for evaluation: The set of decision criteria shall be summarised into sub attributes and attributes, producing evaluation results for quality requirements.

Activity 5 - Conclude the evaluation: In this activity, the software product quality evaluation is concluded, reviewing evaluation results and creating evaluation report.

- Task 5.1 - Review the evaluation result: The evaluator and requester shall review the evaluation results.
- Task 5.2 - Create the evaluation report: Once results are reviewed, the evaluation report is created, including requirements of the evaluation, results from the measurements and analysis performed, any limitations or constraints, evaluators and their qualifications, etc.
- Task 5.3 - Review quality evaluation and provide feedback to the organisation: Evaluator shall review results of the evaluation and validity of the evaluation process, indicators, and metrics applied. Feedback from the review should be used to improve the evaluation process and evaluation techniques.
- Task 5.4 - Perform disposition of evaluation data: When evaluation is completed, the evaluation data shall be disposed according to requirements of the requester, such as returning, archiving or destroying.

In general, software product quality evaluation can be performed during or after the development process or acquisition process by the developer organization, acquirer organization or an independent evaluator. A developer organization can evaluate intermediate software products or final products to ensure that they meet required quality criteria, which can be set by the acquirer or by comparison with other products. The acquirer, on the other hand, can establish quality in use requirements and software quality requirements, can specify requirements to the supplier, can evaluate the software against these requirements before acquisition, and can use the evaluation results to compare alternative products. Finally, independent evaluators can evaluate intermediate software products or final products (requested by a developer, acquirer or some other party) for stakeholders understand, accept, and trust in the evaluation results.

In order to avoid unnecessary work, minimizing specifics challenges or even improving practicality, objectivity, and reliability of the evaluation, it is essential that the implementation of the evaluation process present flexibility to accommodate uniqueness of each application context, such as Systems-of-Systems. Next section presents details about SoS quality attributes and evaluation challenges inherent of this kind of systems.

2.3 Quality Attributes for System-of-Systems

As mentioned early, several quality attributes are critical for SoS (e.g., interoperability, performance, reliability, and security) and, therefore, they should be carefully considered to satisfy the SoS mission (Ackermann et al., 2009). In addition, currently, SoS characteristics can be found in several application domains, such as military, smart home, integrated health systems, crisis management systems, and others. Hence, it is very important the support of a quality model during specification and evaluation of the SoS quality attributes, which are commonly addressed in this variety of domains. However, there is not a clearly decomposition criteria that determines how the complex concept "quality" should be handled for SoS, where constituent systems have complex interdependencies and, sometimes, are developed and maintained by different organizations, with their own stakeholders, development teams, processes, and resources (Gagliardi et al., 2009; Santos et al., 2014). This problem has motivated a growing number of research, since the development of SoS and management of their quality are still a challenge currently, as stated previously by DoD (2008).

In this context, this section presents a panorama about the current state of the art on quality attributes in the SoS context, considering different application domains besides to understand the challenges to evaluate SoS imposed by this context. In this sense, we conducted a SLR, besides to analyze the suitability of the well-established quality model defined by ISO/IEC 25010 regarding the SoS quality attributes that were found. In next section, the main results obtained in our SLR are presented. The research protocol (i.e., the process followed for identifying, analyzing, and extracting information from primary studies) and details about the execution of this SLR are presented in Appendix A.

2.3.1 SoS quality attributes and application domains

In short, during the SLR conduction, 52 primary studies were selected from four different publication databases: ACM Digital Library[1], IEEE Xplore[2], Scopus[3], and Web of Science[4]. These primary studies were used to answer the following research questions (RQ): (i) **RQ1**: *Which are the most common quality attributes for SoS?*; (ii) **RQ2**: *Which are the most common application domains considered for SoS?*; and **RQ3**: *Which are the quality attributes established for each SoS domains?*

First of all, our results indicated that: (i) research on quality attributes for SoS is in increasing during the last years (see Figure 2.4); and (ii) the topic quality attributes for SoS has been investigated by small groups (see Figure 2.5). Analysing the maturity of the research, included studies indicated that the research is overall on a early stage, as most studies were classified as "case study" and "without validation" (see Figure 2.6).

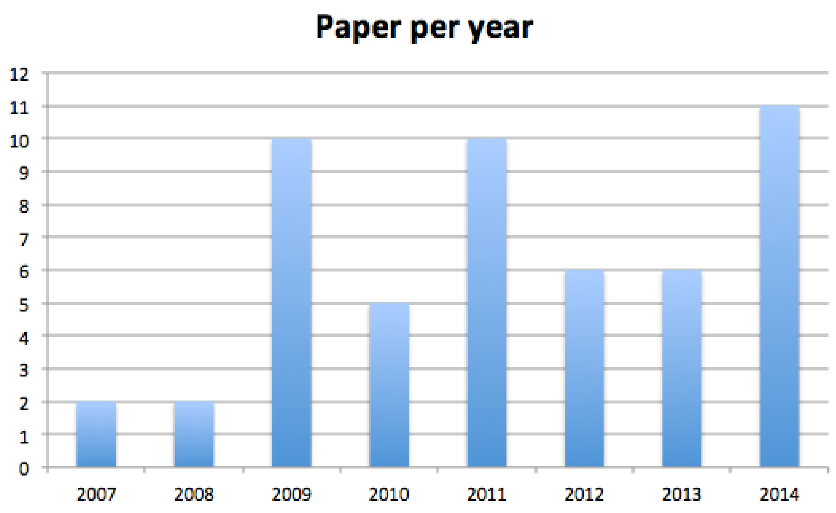


Figure 2.4: Studies included per year

In summary, our results show that the five most relevant quality attributes on SoS research are: security, interoperability, performance, reliability, and safety. Overall, 56 different quality attributes distributed in several domains were identified. Figure 2.7 shows an histogram with all of them and their frequency in the studies considered in this SLR.

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

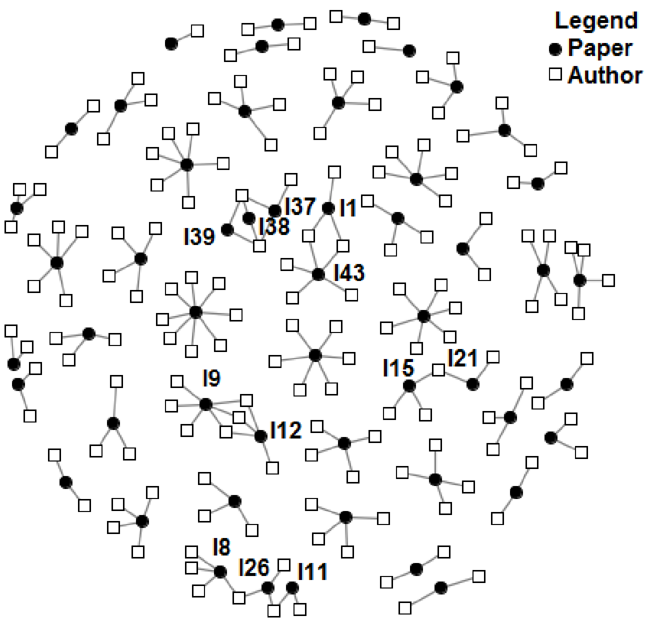


Figure 2.5: Authorship relationship between studies

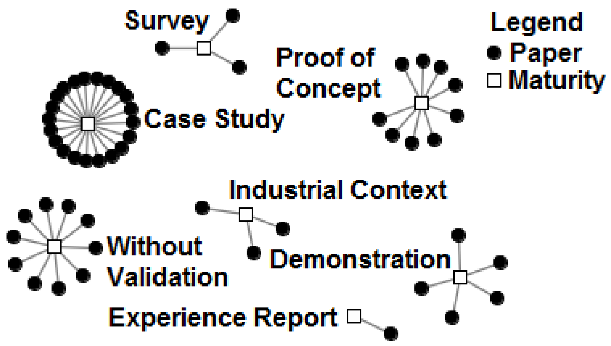


Figure 2.6: Maturity of the included studies

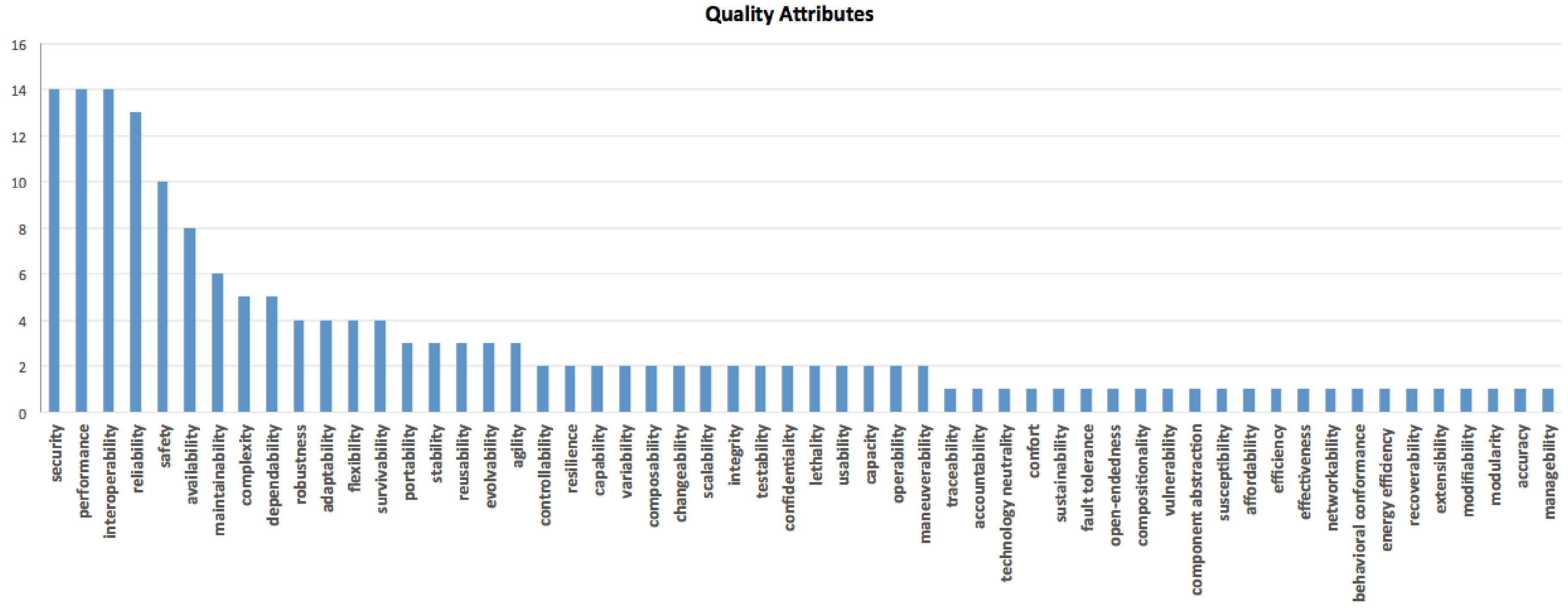


Figure 2.7: Quality attributes identified in the included studies

The military domain is the target of most studies. This is understandable since SoS started to gain their popularity in military domain. However, new application scenarios of SoS have been considered as Figure 2.8 shows highlighting the domains "IT systems", "smart grids" and "automotive". In the IT systems domain, SoS characteristics have been increasingly incorporated, mainly because the integration requirements of these systems with third-party systems (Calinescu et al., 2012; Kimura et al., 2011; Tsadimas et al., 2014). More specifically, Kimura et al. (2011) indicate that this is a tendency due to popularization of cloud computing on the industry. On the other hand, smart grids domain is being investigated in the context of energy management systems, which have as main priority the integration with communication services to detect and address incidents before they compromise the power offering (Singh and Dagli, 2009; Wada et al., 2008; Zafar et al., 2014). Finally, a modern automotive industry is typically driven by the integration of more than 50 embedded computers, also known as ECUs (Electronic Control Units) (Aoyama and Tanabe, 2011; Eklund and Bosch, 2014; Fuchs and Rieke, 2010). More specifically, Fuchs and Rieke (2010) investigate vehicles and roadside units that can communicate in ad hoc way to exchange information, such as safety warnings and traffic information, to avoid accidents and traffic congestion.

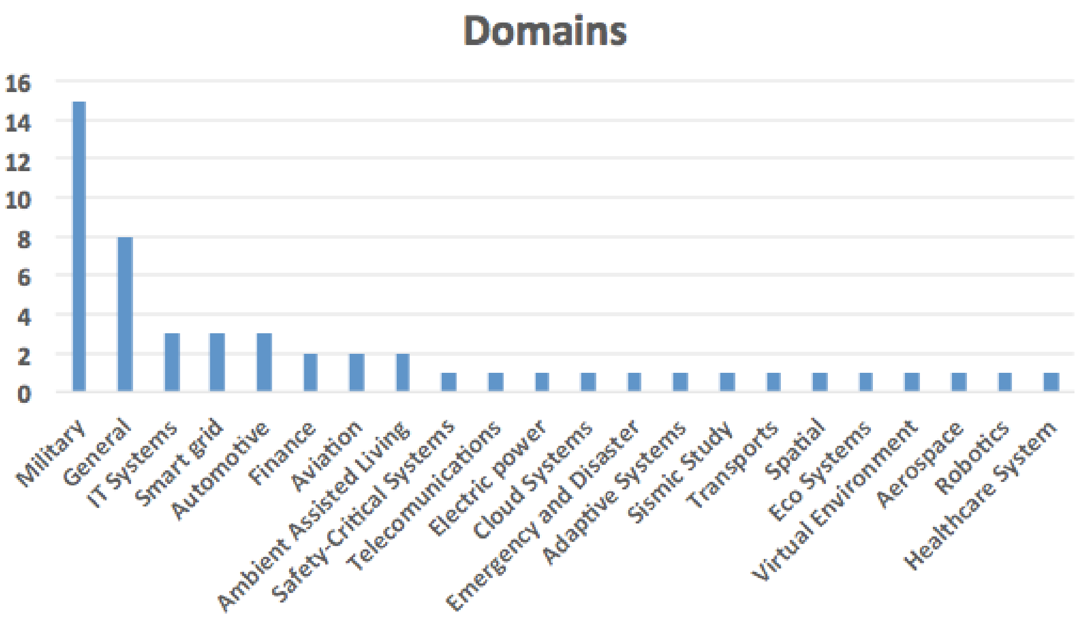


Figure 2.8: Application domains identified on the included studies

Our results also indicate that domains addressed in Figure 2.8 are concerned with a similar set of quality attributes, such as security, interoperability, performance, reliability, and safety. Next section will present a discussion about these five most frequent quality attributes for SoS and their specific evaluation challenges. Besides that, we analyzed how the known quality models, specifically ISO/IEC 25010, can support specification and evaluation of these attributes, considering characteristics and challenges of SoS context.

2.3.2 Discussion and Evaluation Challenges

As mentioned, results suggest that besides military domain, new application scenarios of SoS have emerged, including safety-critical systems, such as military, automotive, and crisis and emergencies management systems. In these situations, SoS must be able to react appropriately to dynamic changes to assure its behavior and quality (Schneider and Trapp, 2011). Therefore, still according to Schneider et al., it is very difficult to assure quality properties like safety, reliability, performance, and security for the whole SoS. The quality of the constituent systems is not enough to ensure the SoS overall quality; therefore, it is necessary to consider the end-to-end quality of the whole SoS. In this context, the dependence, tradeoffs, and relations among quality attributes become more evident and complex.

Another problem found is that some well-established definition for quality attributes, such as for reliability, can not be fully applied in the SoS context. Garro and Tundis (2014) highlight that this problem happens because in SoS the concept of mission failure is not so easily identifiable in comparison to monolithic systems, where failure scenarios and their effects can be clearly identified. Therefore, the SoS reliability is an wider and more flexible concept that should be taken into account considering the flexible and dynamic nature of SoS.

In most SoS domains identified in our SLR, “safety” is an essential quality attribute or even the main goal of these SoS. According to ISO/IEC 25010 (2011), safety is the degree to which a product or system mitigates the potential risk to people in the intended context of its usage. In SoS, these potential risks can become hazardous situations due to failures in any of the constituent systems. This shows that availability, reliability, performance, and security attributes must be properly addressed for each constituent system (Schneider and Trapp, 2011). It is also important to note that these quality attributes and their inter-dependences/relations are not properly considered in the hierarchical structure found in ISO/IEC 25010 and other quality models.

On the other hand, “interoperability” can be handled as a cross-cutting concern that has an unique and crucial coordination role relative to the others quality concerns (Rothenberg, 2008). However, an SoS is usually conceived without considering the interoperability of its constituent systems on the early stages of development (Madni and Sievers, 2014). In addition, metrics established by current quality models can not directly measure interoperability (Guariniello and DeLaurentis, 2014b). This is explained by the emergent behavior of SoS that makes difficult to capture and evaluate interoperability at the level of constituent systems (Meilich, 2006). Hence, it is important to consider the interdependence between quality attributes in scenarios where cascading failures and bottlenecks could result in a complete SoS blackout (Chiprianov et al., 2014). In this sense, quality attributes needs to be measured and controlled for each constituent system to address the impact on SoS quality (Chiprianov et al., 2014; Gorod et al., 2007; Waller and Craddock, 2011).

Moreover, it is necessary to take into account the SoS specific characteristics and how they impact the quality of SoS as a whole. For instance, considering performance and security, some example of the impact of SoS characteristics on these quality attributes are (Chiprianov et al., 2014; Gorod et al., 2007; Waller and Craddock, 2011): (i) the operational independence of the constituents can lead to potential incompatibilities and conflicts between security or performance of each system; then, some systems may be more vulnerable to attacks or have less critical real-time constraints than others and, therefore, their effects and consequences must be carefully considered in the SoS. (ii) due to managerial independence, constituents can have to protect themselves within the SoS from other systems and from SoS emerging activities, which can bring consequences to the SoS mission; (iii) evolutionary development of SoS makes difficult to completely specify quality requirements, such as performance and security at design time, and will need to evolve as the SoS evolves; (iv) emergent behavior difficults the clear identification of the source of a security fault or performance degradation; and (v) geographic distribution makes difficult to achieve security or performance in the SoS as a whole, mainly when different national regulations must be met and when functionalities are met due to numerous and long paths of interaction.

In addition, a first analysis of the ISO/IEC 25010 coverage on the SoS quality attributes shows that 48% of the quality attributes commonly considered in SoS are not addressed by this standard as presented in Table 2.3. So, this can be evidence that, currently, SoS have been developed and mainly evaluated without considering some important quality attributes, since they are not being properly identified and handled by a standardized quality model.

Table 2.3: ISO/IEC 25010 quality attributes coverage

#	Quality Attributes	Among	Covered by ISO
1	performance	14	yes
2	security	14	yes
3	interoperability	14	yes

continued on next page ...

#	Quality Attributes	Among	Covered by ISO
4	reliability	13	yes
5	safety	10	yes
6	availability	8	yes
7	maintainability	6	yes
8	complexity	5	no
9	dependability	5	no
10	robustness	4	no
11	survivability	4	no
12	flexibility	4	yes
13	adaptability	4	yes
14	agility	3	no
15	reusability	3	yes
16	stability	3	no
17	evolvability	3	no
18	portability	3	yes
19	changeability	2	no
20	operability	2	yes
21	lethality	2	no
22	controllability	2	no
23	composability	2	no
24	integrity	2	yes
25	scalability	2	yes
26	testability	2	yes
27	capability	2	no
28	confidentiality	2	yes
29	variability	2	no
30	usability	2	yes
31	capacity	2	yes
32	resilience	2	no
33	maneuverability	2	no
34	open-endedness	1	no
35	susceptibility	1	no
36	extensibility	1	no
37	fault tolerance	1	yes

continued on next page ...

#	Quality Attributes	Among	Covered by ISO
38	networkability	1	no
39	behavioral conformance	1	no
40	vulnerability	1	no
41	confort	1	yes
42	technology neutrality	1	no
43	component abstraction	1	no
44	sustainability	1	yes
45	affordability	1	no
46	traceability	1	no
47	effectiveness	1	yes
48	effectiveness	1	yes
49	energy efficiency	1	yes
50	managebility	1	no
51	accountability	1	yes
52	accuracy	1	yes
53	recoverability	1	yes
54	modifiability	1	yes
55	compositionality	1	no
56	modularity	1	yes

As conclusion, the analysis of the five most relevant quality attributes pointed out limitations of the current quality models, specifically, the well-established and widely used ISO/IEC 25010. SoS quality attributes have also complex interdependencies that can not be translated by a hierarchical structure. Besides that, some well-established definitions for each quality attribute can not be fully applied in the SoS context due to the flexible, dynamic nature of these systems.

2.4 Final Remarks

This chapter introduced, firstly, concepts related to SoS, their characteristics, common application domains, types, and main differences between SoS and monolithic systems. After that, we presented main concepts related to software quality with a special focus on quality evaluation aspects, such as quality models, metrics, evaluation process, techniques and strategies. In addition, an overview about the current quality models and international standards addressed to evaluation of software products was provided. Finally, we presented a SLR about the commonly considered SoS quality attributes and their application domains. From results of this SLR, we provided an important discussion about main five quality attributes found, and current challenges involved in their evaluation considering the SoS context. This information allows to understand our research context and the main contributions of the SoS Evaluation Model proposed by this Master's project in next chapter.

Chapter

3

Systems-of-Systems Quality Evaluation Model

This chapter presents a quality evaluation model suitable to the SoS context. This model has been built in the context of a large international research project entitled RESCUER project[1] that had as main goal to develop a crisis/emergency management SoS.

The remainder of this chapter is organized as follows. Section 3.1 presents the overall structure of the SoS quality evaluation model. Sections 3.2 to 3.6 present in details all activities that compose our model. Finally, Section 3.7 presents final remarks on the topics covered in this chapter.

3.1 Overall Structure

This quality evaluation model intends to cover all important evaluation activities considering all SoS characteristics and challenges not usually addressed by other models. In this sense, evaluation activities defined by current software quality standards and quality models were reviewed and restructured to allow the quality evaluation of SoS.

Figure 3.1 presents an overview of the main activities that compose this SoS Evaluation Model. In summary, it is divided in five activities: (i) Activity 1 - Establishment of the SoS evaluation requirements; (ii) Activity 2 - Specification of the SoS evaluation; (iii)

[1]`http://www.rescuer-project.org`

Activity 3 - Design of the SoS evaluation; (iv) Activity 4 - Execution of the SoS evaluation; and (v) Activity 5 - Conclusion of the SoS evaluation. It can be observed that Activity 4 is decomposed in sub activities conducted in a system level. These sub activities are repeated for each constituent system during all project iterations until no evaluation work is more needed. In next sections, all these activities, sub activities, and artifacts to be produced are detailed.

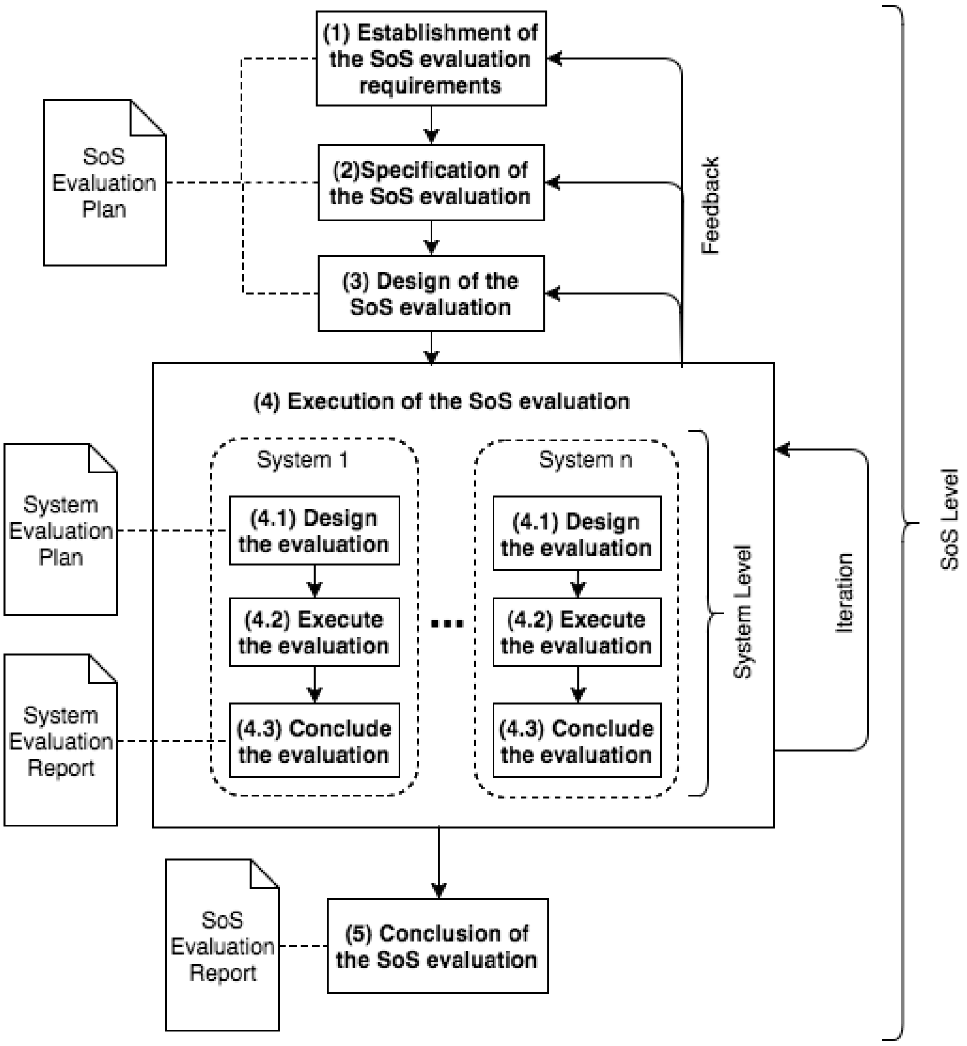

(1) Establishment of the SoS evaluation requirements
SoS Evaluation Plan
(2)Specification of the SoS evaluation
(3) Design of the SoS evaluation
Feedback
(4) Execution of the SoS evaluation
System 1
System n
System Evaluation Plan
(4.1) Design the evaluation
(4.2) Execute the evaluation
(4.3) Conclude the evaluation
System Evaluation Report
System Level
Iteration
SoS Level
SoS Evaluation Report
(5) Conclusion of the SoS evaluation

Figure 3.1: Overall structure of SoS Evaluation Model

To make possible an adequate application of this evaluation model in most SoS contexts, some important information are presented bellow:

- In a SoS, constituent system organizations can adopt different development processes than that used for SoS team. Therefore, their internal quality assurance activities are independently conducted to deliver a level of quality to the SoS. On the other hand, SoS evaluation supported by this evaluation model will ensure that quality attributes important to the mission are achieved in both constituent systems and whole SoS;

- This SoS Evaluation Model can be used by developers, acquirers, quality assurance, control staff, and independent evaluators, particularly those responsible for specifying and evaluating the SoS quality. In this sense, the evaluation activities described in this model can be conducted by both independent evaluation organization teams and members of SoS/constituent systems teams;

- Some activities described in this evaluation model, mainly these ones regarding the SoS evaluation planning (Activities 1, 2, and 3), do not necessarily need to be conducted in sequence. In addition, some information needed to conclude the evaluation planning activities can not be available initially and will be obtained during the project. In this sense, learned lessons and important feedbacks obtained during execution of the evaluation can be used to improve requirements, specification, and design of SoS evaluation;

- The number of evaluation iterations will depend on the each project. Considering the evolutionary and adaptive development commonly found in SoS projects, where new functions, purposes, and quality requirements are added, removed, and modified according to emerging needs, an evaluation iteration could be conducted several times in a SoS project as a way to control the evolution of SoS quality. This would facilitate the quality management of a SoS, since it would allow the gradual quality evaluation of SoS quality requirements. For example, in SoS projects with an iterative development process, an evaluation iteration could be conducted in the final of each project iteration. However, evaluation effort and cost sometimes can limit the number of evaluations conducted in projects with several short iterations, as commonly adopted in agile developments;

- Due to emergent behaviors of SoS, some quality requirements of SoS can not be evaluated only through constituent system evaluations. Hence, it will be needed to consider the whole SoS as an independent system (more details about this are provided during the description of this evaluation model). In this sense, it is important to highlight that, in this SoS Evaluation Model, the term “system” refers to the constituent systems and also to SoS when it is considered as an independent system (see sub activities of Activity 4);

- In general, this evaluation model can be used throughout the lifecycle of any acknowledged SoS, where there are recognized goals, a management team, and resources despite the constituent systems retain their managerial and operational independence; and

- During the SoS evaluation, four different documents are produced: an evaluation plan and an evaluation report for both SoS and system levels. The SoS Evaluation Plan is built based on the output of all SoS evaluation planning activities (Activities 1, 2, and 3) and constantly improved based on the feedback obtained from the conduction of system level activities (sub activities of Activity 4). The System Evaluation Plan is built based on guidelines/requirements defined into SoS Evaluation Plan and restrictions imposed by the constituent system application context. Finally, the System Evaluation Report describes results obtained during the constituent system evaluation, whereas the SoS Evaluation Report presents aggregated results of all constituent system evaluations to provide information of the whole quality of SoS.

The SoS Evaluation Model is based on recommendations and guidelines provided in the main international standards addressed to software quality. Some of these standards were used as base to development of general structure of this evaluation model, whereas others are used as base to the conduction of some evaluation activities. Bellow, it is described the relation of our SoS Evaluation Model with current international standards and quality models.

Firstly, each evaluation activity that composes the evaluation process defined in ISO/IEC 25040 was reviewed and restructured to allow the quality evaluation in SoS level. In the sense, evaluation activities were divided in SoS level (Activities 1 to 5) and system level (sub activities of Activity 4).

In Activity 1 (Establishment of the SoS evaluation requirements), standard ISO/IEC 25010 is used as base to select SoS quality attributes. Hence, the quality model established by this standard was tailored to reflect the crisis and emergency domain. In addition, the structure of ISO/IEC 25010 quality model was adapted to describe quality attributes of all constituent systems in a single structure, and also to describe the importance of each quality attribute to each constituent system.

In Activity 2 (Specification of the SoS evaluation), international standards ISO/IEC 9126-2 - External Metrics and ISO/IEC 9126-4 - Quality in Use Metrics are used as base to define quality metrics needed to measure each quality attribute defined in the established quality model.

In Activity 3 (Design of the SoS evaluation), the evaluation design recommendations presented in ISO/IEC 25040 were raised to cover the schedule of all constituent system evaluations considering the expected number evaluation iterations of a SoS project.

In Activity 4 (Execution of the SoS evaluation), specifically sub activities 4.1, 4.2, and 4.3 are basically established as described in ISO/IEC 25040 process, as they still are conducted in a single system level. However, to allow suitable conduction of these sub activities in the SoS context, guidelines are also provided.

In Activity 5 (Conclusion of the SoS evaluation), the aggregation model defined by Quamoco quality model (Wagner et al., 2012) was used with some adaptation. This is needed to guide the aggregation of evaluation results of each constituent system to obtain the measurement of whole SoS quality, considering the importance of each constituent system in the achievement of SoS mission.

It is noteworthy that for all evaluation activities defined in our model, guidelines are provided to support their suitable conduction, considering the SoS characteristics and challenges found in this context. These guidelines were defined based on our experience in SoS quality evaluation, more specifically in the context of the research project in which this Master's project takes part. In next section, all evaluation activities defined in our SoS Evaluation Model are detailed.

3.2 Activity 1: Establishment of the SoS Evaluation Requirements

Specifically in this activity, the main objective is to document all information needed to guide the evaluation strategies, methods, and techniques that will be used, besides helping to foresee the required effort for the SoS quality evaluation.

Regardless the application context or type of system, the first task to be performed in a quality evaluation is to define its purpose (ISO/IEC 25040, 2011). Basically, the ultimate objective is to ensure that the product provides required quality, meeting stated and implied needs of the users. In the sense, the expected purpose of a SoS evaluation may be, for instance: (i) to assure quality for a SoS and its constituent systems; (ii) to verify the acceptance of a new SoS functionality and decide when to release it; (iii) to assure that a SoS is able to perform its mission; (iv) to access the ongoing feasibility of the project in development; and (v) to predict or estimate final quality of the whole SoS.

Another task in this activity is to identify all constituent systems that will be object of evaluation and also their quality requirements. It is important to obtain the documentation (e.g., requirements specification, design diagrams, and test documentation) of all considered constituent systems. Existing specification of quality requirements provided by organizations involved may be reused, reviewed, and refined during this process. Besides that, quality requirements addressed to the SoS level also need to be identified and/or defined. Therefore, stakeholders of SoS level (e.g., project manager, evaluation requester, constituent project leaders, final users, and requesters of evaluation report) and constituents level (e.g., developers, users, operators, and maintainers) shall be identified and explored to adequately understand the quality requirements in both SoS level and system level.

Using all these information sources, the SoS quality requirements that will be considered in the quality evaluation are specified using a quality model, which is the main output of this activity. The quality model presented by ISO/IEC25010 can be used as basis to the selection of software quality attributes of a specific domain. For selecting quality attributes that will compose the SoS quality model, it is important to consider project constraints, such as evaluation budget, target date for evaluation, and purpose of the evaluation, as all identified quality attributes could not be viable to be evaluated. To obtain a consensus, the involvement of stakeholders is very important. To support this task, a survey can be performed with requirement teams, developers, task leaders, and project coordinators to assure that all selected quality attributes are appropriate and relevant regarding the SoS quality requirements. Moreover, this established quality model can be updated during the SoS development project, by adding new quality attributes not considered previously.

In addition, it is important to define the priority or importance of each quality attribute to be evaluated. This information will also impact the definition of strategies and techniques that will be applied, besides the definition of acceptance criteria that will be used to decide whether the evaluation results are satisfactory or not. This prioritization allows the measurement of whole SoS quality, considering the relative importance of each quality attribute.

Figure 3.2 presents a template to SoS quality model. In this model, it is possible to present quality attributes of all considered constituent systems in a single SoS quality model, together with their respective priorities. For example, for Constituent System 1 (CS1) are considered the Quality Sub Attribute 1.1 as having a middle priority (M) and Quality Sub Attribute 2.2 having high priority (H), whereas for the Constituent System 2 (CS2) are considered Quality Sub Attribute 1.1 having high priority (H) and Quality Sub Attribute 2.1 having low priority (L).

Despite of all, evaluation of each constituent system can not be sufficient to guarantee quality of the whole SoS, mainly because some quality requirements only become evident when constituents are working together. Usually, these quality requirements are result of emergent behaviors of SoS; therefore, they can not be measured through only of constituent evaluations. Therefore, to also support the evaluation of specifics SoS quality requirements, it is needed to consider SoS like an independent system/a black box system. As the example in Figure 3.2, Quality Sub Attribute n.1 has high priority (H) and

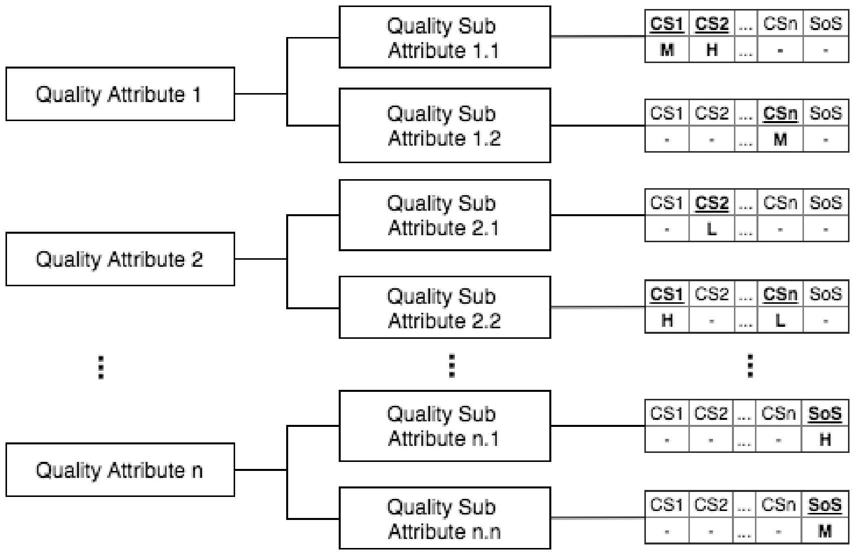

Quality Attribute 1
Quality Attribute 2
Quality Attribute n
Quality Sub Attribute 1.1
Quality Sub Attribute 1.2
Quality Sub Attribute 2.1
Quality Sub Attribute 2.2
Quality Sub Attribute n.1
Quality Sub Attribute n.n
CS1 CS2 ... CSn SoS
M H ... - -
- - ... M -
- L ... - -
H - ... L -
- - ... - H
- - ... - M

Figure 3.2: Structure of SoS quality model

Quality Sub Attribute n.n has middle priority (M) to SoS and, therefore, the whole SoS will be measured in a specific evaluation. Evaluation of an SoS as an independent system is detailed in next sections.

3.3 Activity 2: Specification of the SoS Evaluation

The main goal of this activity is to define the quality metrics to each quality attributes (of the quality model previously established) as well as their acceptance criteria, which are numerical thresholds or targets used to determine the need for action or further investigation, or to describe the level of confidence in a given result (ISO/IEC 25040, 2011).

It is important to highlight that, in the SoS context, different quality metrics, measure strategies, and acceptance criteria could be defined for each constituent system considering its resources, measurement context, and evaluation purpose. This scenario is more common in an SoS with high level of managerial independence of its constituent systems, or when different organizations are responsible by the conduction and report of the constituents quality evaluation. Besides that, it is important to remember that suitable quality metrics could also need to be defined to measure quality attributes addressed to the SoS level.

To define quality metrics to each quality attribute in the established quality model, international standards, such as ISO/IEC 9126-2 - External Metrics and ISO/IEC 9126-4 - Quality In Use Metrics, can be used as base. Quality metrics proposed by these standards can be selected and tailored to the considered application context. The choice of the methods or techniques that will be used to obtain input data to each metric, depends on the importance and relevance of the quality attribute to achieve SoS mission. In particular, the quality attribute priorities defined in the SoS quality model (low, middle, and high are used as example, but any prioritization rank could be adopted) may be used as a guide to select evaluation techniques. For instance, evaluation techniques suggested by ISO/IEC 25040 can be linked to the quality attribute priorities, considering the needed rigor in each evaluation, as presented in the Table 3.1.

Table 3.1: Evaluation techniques vs quality attributes priorities

Quality Attribute	**Priority**	**Evaluation Techniques**
Functionaly	L	Functional or black box testing
	M	Inspection of development documentation guided by checklists
	H	Unit testing with test coverage criteria
Reliability	L	Verification of the use of specific programming language facilities
	M	Analysis of fault tolerance construct in the software design and code
	H	Reliability growth modeling
Usability	L	User interface and documentation inspection
	M	Verification of the conformity to interface standards
	H	Performing usage experiments with real users
Efficiency	L	Execution time measurement
	M	Benchmark testing
	H	Analysis of the design to determine the algorithmic complexity
Maintanability	L	Inspection of development documentation guided by checklists
	M	Code measures and programming rules verification
	H	Analysis of traceability between elements of development documentation
Portability	L	Analysis of software installation procedures
	M	Programming rules verification
	H	Analysis of software design

Based on that, the SoS team or independent organizations can suggest the methods/techniques/tools more suitable to obtain information inputs to the metrics. However, the selection of these evaluation techniques will also depend on each constituent system context. Hence, detailed specification about which and how these strategies will be applied in the evaluations should be preferably defined with a broad participation of each involved organization (Sub Activity 4.1 - Design the evaluation in system level), as it will depend on several variables, e.g., available resources and internal process, which can not be clearly known by SoS team or independent evaluation organizations. This detailing will compose the System Evaluation Plan. In the same way, a System Evaluation Plan also needs to be developed to SoS when it is considered as an independent system. Details about this artifact will be presented in Section 3.5 and Section 4.

Ideally, the acceptance criteria are defined from the SoS specification and SoS quality requirements since it will be used as reference to the system quality determination. However, this information could not be still available, in the needed detail, in the current project iteration. The SoS team can lead the elicitation of these criteria from the involvement of stakeholders, requirement teams, developers, task leaders, project coordinators, domain experts, among others, to obtain consensus about quantitative targets for each quality attribute. Next section presents examples of these quantitative targets.

3.4 Activity 3: Design of the SoS Evaluation

The main goal of this activity is to plan and design the SoS evaluation. Evaluation of the quality attributes of all constituent systems as well as those defined specifically to the SoS are scheduled. The main concern is to define which quality attributes and metrics will be considered for each system and in which evaluation iteration. In addition, it is important to define people, constituents, and external organizations that will be responsible to execute and report each expected evaluations.

This resulting plan must also take into account availability of resources from the constituent organizations and SoS team, besides quality requirements dependencies. For example, a constituent organization or SoS team could not evaluate a specific quality attribute or apply a specific quality metric in an evaluation iteration due to the development status or lack of resources. Besides that, due to the evolutionary development of SoS, design of SoS evaluation produced in this activity must be updated or revisited constantly after starting evaluations, since some items of the evaluation plan could have been only defined at a high level in the early phase of the project.

To support the designing of SoS evaluation, Table 3.2 can be used as a model to define which quality attributes and metrics will be considered for each evaluation in which iteration. In addition, the acceptance criteria defined in last activity (Activity 2) can be fractioned to obtain the target value in the final project iterations. In the example presented in Table 3.2, Metric 1 (M1) used to measure Quality Sub Attribute 1.1 (QSA1.1) is applied to Constituent System 1 (CS1) in the project interaction 1, 2, and N with increasing acceptance criteria (50% in iteration 1, 85% in iteration 2, and 100% in iteration N. Defining 100% as acceptance criteria in a project iteration means that all quality requirement should be achieved in this phase). M2 (also used to measure QSA1.1) is applied to both CS1 and CS2. However, CS2 is only considered from the iteration 2 onwards. On the other hand, M3 is considered only to CS2 in the iteration 1 and N. Evaluation of SoS as an independent system is scheduled to the last project iteration, considering the metric Mn used to measure the Quality Sub Attribute n.n (QSAn.n)

The main outputs from this activity, as well as the outputs from last two activities (defined as SoS planning activities) will compose a document defined as SoS Evaluation Plan. This document will be used as basis to all evaluations conducted in system level (detailed in next section). Moreover, it can be constantly improved based on the feedback obtained from the conduction of these evaluations. In general, the SoS Evaluation Plan

Table 3.2: SoS evaluation plan example

Quality Attribute.	**Quality Sub Attribute.**	**Metric**	**Iteration 1**		**Iteration 2**		**Iteration N**	
			Const. System	**Accept. Criteria**	**Const. System**	**Accept. Criteria**	**Const. System**	**Accept. Criteria**
QA1	QSA1.1	M1	CS1	50%	CS1	85%	CS1	100%
		M2	CS1	80%	CS1	90%	CS1	100%
			-	-	CS2	75%	CS2	100%
QA2	QSA2.1	M3	CS2	70%	-	-	CS2	100%
...	...	...	...	...	...	...	...	...
QAn	QSAn.n	Mn	-	-	-	-	SoS	100%

will contain information such as: a) SoS overview; b) information about the constituent systems; c) SoS Evaluation purpose; d) SoS quality model, quality metrics, acceptance criteria, and evaluation methods/tools; e) SoS evaluation design considering the project iterations and all constituent systems; f) organisations involved in the evaluation, such as an independent/external evaluation organisations, constituent systems organizations, client organizations, together with their responsibilities; g) evaluation budget issues; h) schedule for the evaluation milestones; i) evaluation context and environment to be considered; and j) required standards to be adopted in the evaluation.

3.5 Activity 4: Execution of the SoS Evaluation

The execution of the SoS evaluation is decomposed in a set of sub activities conducted in the system level. These sub activities (4.1-Design of the Evaluation, 4.2-Execution of the Evaluation, and 4.3-Conclusion of the Evaluation) are repeated for each system during the project iterations. Since these activities are applied in a single system level, they can be executed as suggested by ISO/IEC 25040; however, some consideration are necessary.

The design of each system evaluation needs to be conducted according to the SoS Evaluation Plan, which defines which quality attributes, metrics, methods/tools, and acceptance criteria must be considered in each expected evaluation, considering all the foreseen iterations. In particular, the System Evaluation Plan needs to present, in a detailed way, all evaluation activities, techniques, tools, and methods that will compose the evaluation strategy specific for a given system. It is important to remember that these evaluation strategies needs to be able to produce input data to application of each metric defined in SoS Evaluation Plan and, consequently, enable the measurement of each quality attribute. In the same way as defined to constituent systems, in the case of an evaluation considers the SoS as an independent system, a System Evaluation Plan also need to be produced detailing how exactly this evaluation will be conducted.

In summary, a System Evaluation Plan should contain information such as: a) SoS overview; b) system considered in the evaluation; c) system evaluation purpose; d) quality attributes, metrics, and acceptance criteria to be considered; e) evaluation strategy, method and tools to be used; f) organizations or people to be involved in the evaluation together with their responsibilities; g) evaluation context and environment to be considered; and h) required standards to be adopted in the evaluation.

After executing the evaluation, in the Activity 4.3 - Conclude the evaluation, evaluation results are compared with acceptance criteria defined in the SoS Evaluation Plan. This enables to identify if the systems meet the quality requirements regarding the current evaluation iteration. These results are reported into a System Evaluation Report. In general, depending on how the evaluation report is intended to be used, it should include the following items: a) summary of System Evaluation Plan; b) evaluation results from the measurements and analysis performed; c) quantitative analysis from comparison between metric results and acceptance criteria; d) limitations, constraints, deficiencies, or exclusions in an evaluation activity, including their impact on the use, configuration, modification, decision taken, or general maintenance of the system over time; e) the evaluators and their qualifications; and f) other information necessary to be able to repeat or reproduce the evaluation.

From the evaluation conduction experience and analysis of evaluation results, it is possible to identify: (i) system deficiencies or evaluation limitations; (ii) need of any additional evaluations; (iii) identification of new quality requirements or update of their priority; and (iv) suitability of the evaluation strategies and methods used. This feedback must be used to improve the SoS Evaluation Plan that will be the base to subsequent constituent evaluations during next evaluation iterations.

3.6 Activity 5: Conclusion of the SoS Evaluation

In this activity, results of each system evaluation are put together to obtain the measurement of the whole quality of SoS. This measurement can be done in the final of each evaluation iteration as a way to control the evolution of SoS quality, through its gradual measurement. Firstly, the quality of each constituent system is obtained through the composition of the measurement result of each quality attribute. Considering that each quality attribute has different priorities for each constituent system, the total quality needs to consider different weights to each quality attribute and sub attributes.

After obtaining the total quality measurement of each constituent system, a similar approach can be applied to obtain the quality of the SoS, considering different weights for each constituent system that composes the SoS. This needs to be considered, as each constituent system can have different priorities and importance in the achievement of the SoS mission.

To guide this quality aggregation an aggregation model to SoS quality evaluation was defined based on that proposed by Quamoco quality model (Wagner et al., 2012). This quality model proposes that in a hierarchical quality model, the measurement of whole quality is calculated by the weighted sum of the measurements of all sub-factor. This quality aggregation is successively conducted to next level of quality model until the total product quality be obtained. According Wagner et al. (2012), the aggregation operator defined as a weighted sum is an easily understandable and reliable aggregation approach.

Figure 3.3 presents the SoS quality aggregation model defining composition levels needed to obtain the measure of the whole quality of SoS. Firstly, results from each metric are aggregated to obtain the quantitative value that refers to the degree of compliance to the corresponding quality attribute. This aggregation step is repeated for all quality measures of each constituent system. Finally, SoS quality is obtained through the aggregation of quality measure of all its constituent systems.

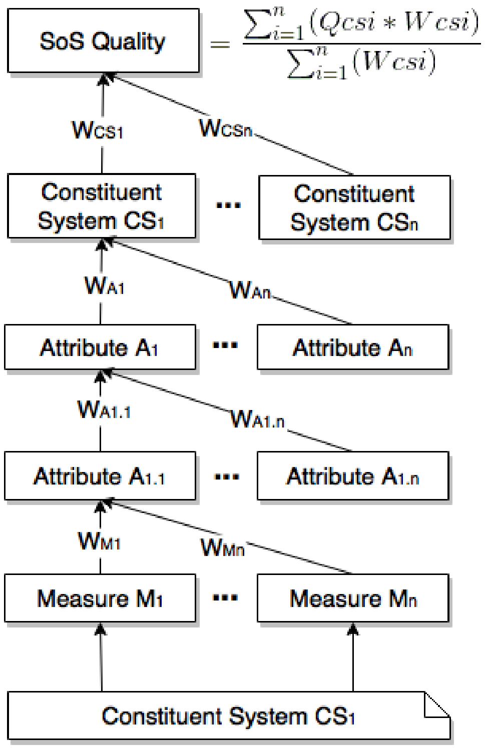


Figure 3.3: SoS aggregation model

For example, consider a SoS with three constituent systems: CS1, CS2, and CS3 with importance/priority defined as H, M, and L, respectively, by stakeholders regarding the achievement of SoS mission. The weights (w) assigned to H is 3, M is 2, and L is 1. Suppose that during the constituent systems evaluations, the quality (Qcs) obtained were 0.7 to CS1, 0.85 to CS2, and 0.60 to CS3. From this information, it is possible to calculate the SoS quality (QSoS) by the weighted sum using Equation 3.1.

$$Q(SoS) = \frac{\sum_{i=1}^{n}(Qcsi * Wcsi)}{\sum_{i=1}^{n}(Wcsi)} \tag{3.1}$$

After the calculus, it is obtained the Q(SoS) = 0.73 that represent the whole quality of the SoS obtained from the aggregation of the quality of its constituent systems. It is important to highlight that a similar approach is conducted in all phases defined in the SoS quality aggregation model.

As suggested in Figure 3.3, although not mandatory, a weight could be assign to each element in all aggregations levels established in aggregation model. This means, the definition of weights, if possible, to metrics, quality sub attributes, quality attributes, and finally, the weight of each constituent systems. In this context, it is very important to take into account the experts opinion, considering different perspectives (e.g., managers, developers, users), as it produces more trustable results and reflects more precisely the common view of quality of the set of stakeholders (Villalba et al., 2010). These opinions can be obtained, as in the example before, by collecting data with questionnaires using the average values from a rating scale such as L - Low, M - Medium, and H - High, which can be directly translated in weights.

As suggested by Wagner et al. (2012), another way to support the definition of weights is to use the Rank-Order Centroid method (Barron and Barrett, 1996) to calculate the weights of each factor automatically based on a relevance ranking between sibling quality attributes using the Swing approach (Edwards and Barron, 1994). It is needed to rank the set of quality attributes by stakeholders and experts to identify their relative importance. Then, the weight of each quality attribute is calculated by Equation 3.2, considering n as the number of factors (metrics/quality attributes/constituent systems) considered in the rank, and i the position of the factor in the rank.

$$Wi = (1/n)\sum_{j=i}^{n}(1/j) \tag{3.2}$$

Finally, it is important to remember that, in a SoS context, new behaviors and functionalities emerge from the collaboration of constituent systems. As mentioned in Sections 3.2 and 3.3, it is needed to define quality attributes and/or metrics that will be applied to the SoS level as an independent system. Therefore, two evaluation results of SoS quality are produced, one obtained from the aggregation of the evaluation results of all constituent systems, and other from evaluation of SoS as an independent system. This makes it possible to identify problems in some quality attributes that only become evident when constituents are working together. On the other hand, the composition result can help to identify a problem in a constituent quality that has highly impacted the whole quality of SoS. In summary, both results are equally important and complementary.

3.7 Final Remarks

This chapter presented a quality evaluation model suitable to the SoS context. The main goal of this model is to cover important evaluation activities, considering SoS characteristics and challenges. For that, the SoS Evaluation Model was built considering recommendations and guidelines provided in the main international standards addressed to software quality. Most important software quality evaluation activities were reviewed and restructured in both, constituent system and SoS levels. This allowed the understanding of the roles and responsibilities of an SoS evaluation team in the context of an acknowledge SoS. In addition, it allows to understand how system level evaluation activities can be distributed and controlled to a better measurement and management of SoS quality through the evaluation of its constituent systems. In addition, guidelines and recommendations were provided for all evaluation activities defined in this model to allow a suitable conduction in SoS context.

In summary, the proposed model covers five evaluation activities, such as: (i) Activity 1 - Establishment of the SoS evaluation requirements; (ii) Activity 2 - Specification of the SoS evaluation; (iii) Activity 3 - Design of the SoS evaluation; (iv) Activity 4 - Execution of the SoS evaluation; and (v) Activity 5 - Conclusion of the SoS evaluation. During the application of our model, four different documents are produced: (i) SoS Evaluation Plan is built based on the output of all SoS evaluation planning activities and used as basis to planning of all system evaluations. (ii) System Evaluation Plan is built based on guidelines/requirements defined into SoS Evaluation Plan and used as basis to conduction of a specific system evaluation. Finally, (iii) System Evaluation Report describes results obtained during the constituent system evaluation, whereas (iv) SoS Evaluation Report presents aggregated results of all constituent system evaluations to provide information about the whole quality of SoS.

Chapter

4

Case Study: Crisis and Emergency Management Domain

This chapter presents the application of the SoS Evaluation Model described in Chapter 3 in the context of RESCUER project[1]. Our experience regarding the planning, execution, and conclusion of evaluations conducted in the crisis/emergency management SoS developed in this project were previously and partially reported in the papers entitled “Reporting an experience on the establishment of a quality model for Systems-of-System” (Santos et al., 2015a) and “Evaluation of a Crowdsourcing System: An experience report” (Santos et al., 2015b).

The rest of this chapter is organized as follows: Section 4.1 describes our application context; Section 4.2 presents the establishment of the SoS evaluation requirements; Section 4.3 describes the specification of the SoS evaluation; Section 4.4 presents the design of the SoS evaluation; Section 4.5 describes the execution of the SoS evaluation; Section 4.6 presents the conclusion of the SoS evaluation; and finally, Section 4.7 presents final remarks on topics covered in this chapter.

[1]`http://www.rescuer-project.org`

4.1 Application Context

The main challenge for an emergency command and control centre (C&CC) is to quickly obtain contextual information to answer an emergency and ensure correct decisions. An appropriate response is essential to attenuate occurrence of physical injuries as well as negative outcome to the public image of the involved organizations. Decisions based on incorrect or late information have a great potential to cause more damage (Villela et al., 2013).

In parallel, the daily use of mobile devices, such as tablets and smartphones, provides an enabling technology for building new software solutions. Exploring such devices as a communication mechanism, the RESCUER research project proposes the development of an interoperable computer-based solution to provide C&CC with real-time contextual information related to the emergency situation in large-scale events and industrial areas/-chemical parks. This solution relies on the collection, combination, and aggregation of crowdsourcing information of these contexts (Villela et al., 2013).

Large-scale events are public events attended by a lot of people. Public events address every interested visitor who intends to attend to the activities that are offered, for instance, musical performance, sports, or other social activities. Chemical parks are industrial areas in which, among other things, chemicals are stored and processed. In case of an incident in these industrial areas, chemicals can harm employees, civilians in the affected community, and the environment. In Brazil, chemical companies are usually placed in industrial parks. These industrial parks have several companies that must be according to the Brazilian laws for emergency management.

In summary, the RESCUER SoS comprises four main constituent systems (Villela et al., 2013):

- Mobile Crowdsourcing Solution (**MCS**, also RESCUER application or app) implements suitable context-sensitive mechanisms for eyewitnesses and operational forces carrying mobile devices to provide the C&CC with information about emergency situations. This application takes in consideration the behavior of people under stress situations as emergency situations that trigger very basic human instinctive behavior, making people overwhelmed and confused. Users provide reports of incidents with text, photos, and videos. Besides, the RESCUER application is able to send relevant information automatically from device sensors without the necessity of any user interaction, such as user location, direction, velocity, etc.;

- Data Analysis Solution (**DAS**) is composed by algorithms that process and filter received data (e.g., image, text, and video) to extract relevant and consolidated information. This solution is responsible for fusing similar data coming from different eyewitnesses as well as analysing photos, videos, and text messages to extract information, such as the type of incident, position and dimensions of the affected area, people density, surrounding sources of further danger, evacuation routes, and possible approach routes for the first responders;

- Emergency Response Toolkit (**ERTK**) provides the C&CC with updated and relevant information, in an adequate format, to support decision-making during an emergency. It applies a set of solutions to manage the analyzed data coming from DAS and presents them in a real time dashboard, using adequate visualization means; and

- Communication Infrastructure (**COM**) supports the information flow between stakeholders (i.e., eyewitness, workforces, supporting forces) even when traditional communication infrastructure (i.e., WiFi, 3G, 4G, etc.) is overloaded, by establishing an ad hoc network communication to propagate data between users phones and the C&CC.

It is important to highlight that all these constituent systems are completely distributed, as they are developed and maintained by different organizations from four different countries (namely Germany, Spain, Brazil, and Austria), with their own stakeholders, development teams, processes, and resources. The solution that emerged from integration of these constituent systems is considered a SoS, since it can be perceived the SoS characteristics.

These constituent systems are part of an integrated solution entitled RESCUER SoS, which gathers crowdsourcing information from thousands of MCS users, providing relevant information through DAS to the C&CC. Due to criticality of RESCUER SoS, the COM helps to ensure the availability of whole SoS. RESCUER SoS can be considered an acknowledged SoS, since there are recognized goals, a management team, own resources, and at the same time, the constituent systems retain their managerial and operational independence. Figure 4.1 shows an overview of RESCUER SoS and its constituent systems.

An important characteristic of this SoS is also its evolutionary and adaptive development, where structures, functions, and purposes are added, removed, and modified according to emerging needs. To manage expected changes in its requirements, RESCUER project adopted an iterative development process, in which each subsequent iteration builds on and improves results of the previous one. The overall strategy divided the process in three iterations. Basically, iterations were defined according to the integration of functionality (basic functions first, more complex functions later). This facilitated quality

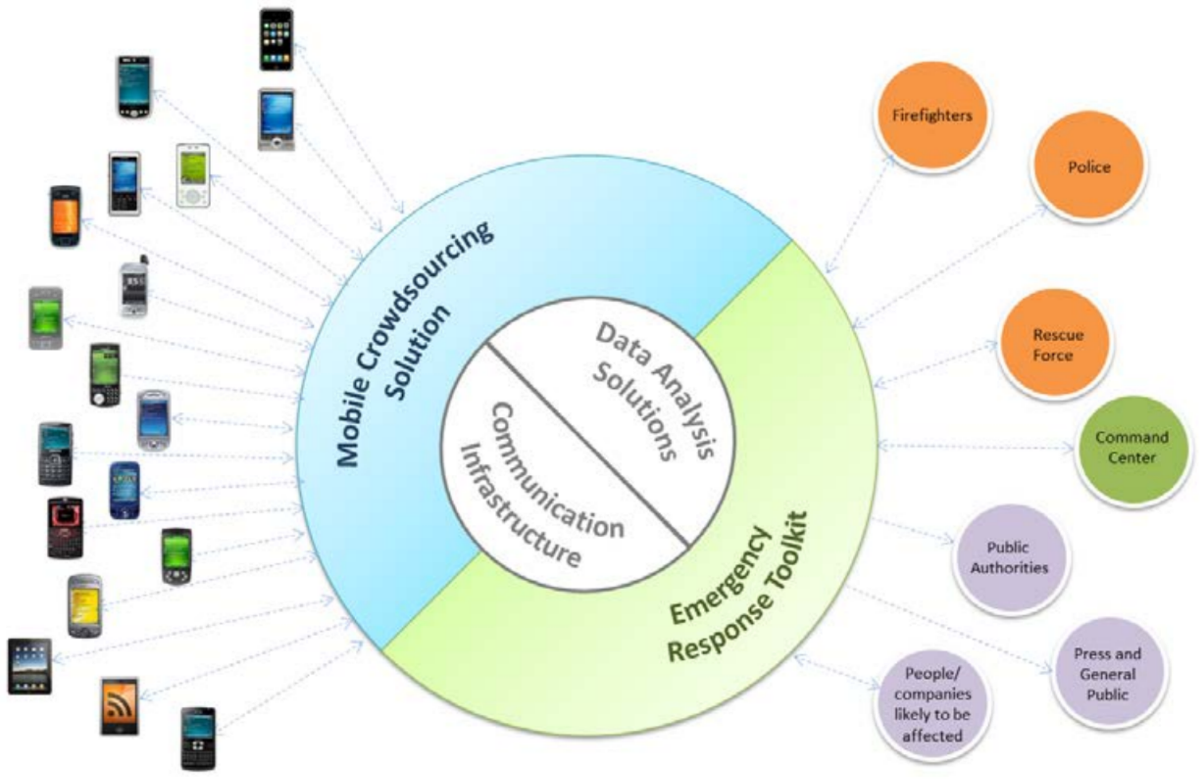

Mobile Crowdsourcing Solution
Data Analysis Solutions
Communication Infrastructure
Emergency Response Toolkit
Firefighters
Police
Rescue Force
Command Center
Public Authorities
Press and General Public
People/ companies likely to be affected

Figure 4.1: RESCUER SoS overview (Villela et al., 2013)

management, since it allows the quality evaluation of first results, and gradual specification and maturation of the requirements of RESCUER SoS. Therefore, RESCUER SoS evaluations were also divided in three iterations, conducted in the final of each project iteration.

4.2 Activity 1: Establishment of the SoS Evaluation Requirements

As early explained, the main output of this activity is the quality model related to the SoS to be evaluated. This quality model was initially based on quality attributes defined in ISO/IEC 25010. To determine which quality attributes and sub attributes were relevant to the RESCUER SoS, all RESCUER non-functional requirements as well as the project goals and scope were carefully identified and analyzed from available documentation. This analysis allowed to translate each non-functional requirement into ISO/IEC 25010 quality attributes, taking into account the product quality and quality in use models. To support this activity, a survey[2] with requirement teams, developers, task leaders, and project coordinators was performed to assure that all selected quality attributes were ap-

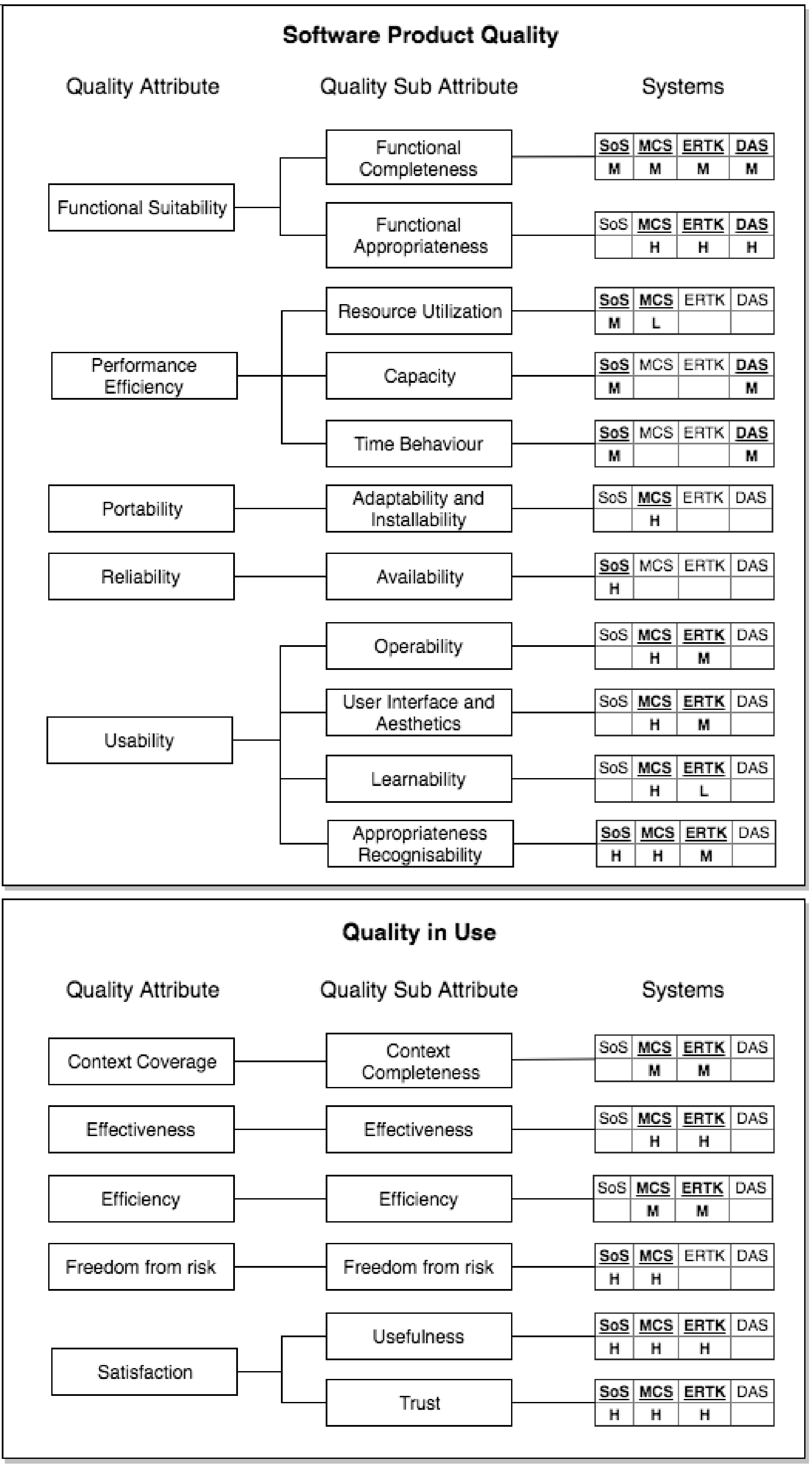
[2] A model of the questionnaire used in this survey can be found in: https://goo.gl/Zi3ulI

propriate and relevant regarding RESCUER SoS requirements. Additionally, suggestions of other quality attributes that could be considered in the quality model were obtained. After questionnaires were answered, a meeting was performed with stakeholders to discuss results and, consequently, to obtain consensus about elements that would compose the quality model.

The involvement of stakeholders was very important, since the requirements about RESCUER SoS were still being detailed in current phase of the project. Therefore, some quality attributes could still not be directly translated from RESCUER requirements. Moreover, this strategy allowed to obtain a consensus about all elements that composed the quality model, besides to assure that main decisions about the quality model were coherent with system requirements and project goals.

To support the evaluation of specific SoS quality requirements that only become evident when constituent systems were working together, some of identified quality attributes were addressed to RESCUER SoS as an independent system. Besides that, it is important to highlight that, not all quality attributes and sub attributes were relevant for all constituent systems and as well as the SoS. Depending on use purpose of the quality model (system specification or evaluation), and evaluation subject, a different subset of attributes/sub attributes could be chosen accordingly to specific goals and objectives. In addition, as RESCUER was an iterative research project, its requirements were constantly modified, and hence, the quality model also was constantly adapted. Therefore, quality attributes that were not considered in initial project iteration were added in following project iterations when needed.

Figure 4.2 presents quality attributes and sub attributes that composed this quality model. In addition, on the right side of figure, it is specified the set of constituent systems that each quality sub attribute was related, and the respective importance that these quality sub attributes had in each evaluation subject. The importance of each quality attribute was defined as H - High, M - Medium, and L - Low. H means that a high rigor should be applied in the definition of evaluations strategies to obtain highly reliable results, and also, a high acceptance criteria should be applied to decide whether the evaluation results were satisfactory or not. On the other hand, L means that a low rigor would be applied in the definition of evaluation strategies and acceptance criteria considered. In summary, this link among quality attributes, priority, and constituent systems was very important in this SoS context, since the priority, way to lead, and cost to evaluate the quality attributes were different to each constituent system.



Software Product Quality

Quality Attribute	Quality Sub Attribute	Systems			
		SoS	MCS	ERTK	DAS
Functional Suitability	Functional Completeness	M	M	M	M
	Functional Appropriateness		H	H	H
Performance Efficiency	Resource Utilization	M	L		
	Capacity	M			M
	Time Behaviour	M			M
Portability	Adaptability and Installability		H		
Reliability	Availability	H			
Usability	Operability		H	M	
	User Interface and Aesthetics		H	M	
	Learnability		H	L	
	Appropriateness Recognisability	H	H	M	

Quality in Use

Quality Attribute	Quality Sub Attribute	Systems			
		SoS	MCS	ERTK	DAS
Context Coverage	Context Completeness		M	M	
Effectiveness	Effectiveness		H	H	
Efficiency	Efficiency		M	M	
Freedom from risk	Freedom from risk	H	H		
Satisfaction	Usefulness	H	H	H	
	Trust	H	H	H	

Figure 4.2: RESCUER quality model

4.3 Activity 2: Specification of the SoS Evaluation

The main output of this activity is the set of quality metrics that will be used to measure all quality attributes as well as their corresponding acceptance criteria, and suggestions of methods/techniques/tools more suitable to obtain information inputs to quality metrics application. When applying a quality metric, it is possible to obtain a quantitative value that characterizes the degree of compliance of a software to the corresponding quality attribute. For each quality sub attribute defined in the quality model, a set of appropriated metrics was established. These metrics were selected and adapted from ISO/IEC 9126-2 - External Metrics and ISO/IEC 9126-4 - Quality In Use Metrics. External metrics usually measure the quality of a software product by measuring the behavior of system, during testing stages or system operation (ISO/IEC 9126-2, 2003). On the order hand, quality in use metrics are applied in a realistic system environment to verify if a product meets the needs of specified users to achieve their goals (ISO/IEC 9126-4, 2003).

Table 4.1 and Table 4.2 present a total of 23 metrics that we established in our quality model. For example, metrics UL1 and UL2 were used to measure the learnability of MCS, a key quality attribute, since no training material should be necessary for the user to understand and interact with the app during an emergency incident, even when users are under high stress situations. Therefore, these metrics are important to identify the influence of demonstration or tutorial in the effectiveness of users and, consequently, to measure the level of learnability of MCS. On the order hand, PT1 was defined to verify the performance in a SoS level of RESCUER SoS, considering all constituent systems working together. This metric measures the average time that a report sent by MCS user takes to be analyzed by DAS and showed to C&CC by ERTK constituent system. This time needs to be carefully controlled for that decision making activities and the suitability of contextual information not be affected.

Table 4.1: RESCUER product quality metric

Quality Attributes	**Quality Sub Attributes**	**Metric**	**Purpose of the metric**	**Method of application**	**Artefact**
Functional Suitability	Functional Completeness	FF1	How complete is the system from the user point of view?	Count the number of user that think the system complete and compare with the total number of users in the evaluation.	Conduct user test and interview user with questionnaires.
	Functional Appropriateness	FF2	How adequate are the evaluated functions of the user point of view?	Count the number of functions in which problems was detected in the evaluation and compare with the number of evaluated function.	Conduct user test and interview user with questionnaires.

continued on next page ...

Quality Attributes	**Quality Sub Attributes**	**Metric**	**Purpose of the metric**	**Method of application**	**Artefact**
Performance Efficiency	Resource Utilization	PR1	How adequate is the battery consumption?	Measure the battery consumption for a specific period of time and compare with the acceptable battery consumption for the same period of time.	The battery status before and after the evaluation
		PR2	How adequate is the processor consumption?	Measure the processor consumption for a specific period of time and compare with the acceptable processor consumption.	The processor status before and after the evaluation
		PR3	How adequate is the memory consumption?	Measure the memory consumption for a specific period of time and compare with the acceptable memory consumption.	The memory status before and after the evaluation
	Capacity	PC1	What is the rate of incident reports that system can handle without problems?	Send the expected load of incident reports and measure the rate that the system can handle.	Load Test (Apache JMeter)
	Time Behaviour	PT1	What is the average time that a report sent from MCS take to be presented by ERTK?	Simulate an expected load of users, and measure the average time to reports be presented by ERTK.	Load Test (Apache JMeter)
Usability	Operability	UO1	How many steps the user need for perform a specific task?	Count the number of steps the user need for perform a specific task and compare with the minimal number of needed interactions	Conduct user test and observe user behaviour
		UO2	What proportion of user think the system easy to use and navigate?	Count the number of user that think the system is easy to use and compare with the total number of users in the evaluation	Conduct user test and interview user with questionnaire
	Appropriateness Recognisability	UA1	What proportion of the product functions will the user be able to understand correctly?	Count the number of user interface functions that are easily understood by the user and compare with the number of functions available for user	Conduct user test and interview user with questionnaires or observe user behaviour.
	User interface aesthetics	UU1	How attractive is the interface to the user?	Measure the score assigned by users to defined criteria in the applied questionnaire	AttrakDiff quest. to assess the attractiveness of the interface to users after experience of usage
	Learnability	UL1	What proportion of user can operate successfully a function without a demonstration or tutorial?	Count the number of users that adequately operated the functions without a demonstration and compare with the total number of users in the evaluation who did not have a demonstration or tutorial	Conduct user test and interview user with questionnaires or observe user behaviour.

continued on next page ...

Quality Attributes	**Quality Sub Attributes**	**Metric**	**Purpose of the metric**	**Method of application**	**Artefact**
		UL2	What proportion of user can operate successfully a function after a demonstration or tutorial?	Count the number of users that adequately operated the functions after the demonstration and compare with the total number of users in the evaluation who had a demonstration or tutorial.	Conduct user test and interview user with questionnaires or observe user behaviour.
		UL3	What proportion of user can learn to use the functions with an acceptable number of attempts?	Count the number of users that can learn to use the functions correctly with an acceptable number of attempts and compare with the total number of users in evaluation	Conduct user test and observe user behaviour.
		UL4	How frequently does an user ask for explanation to learn to operate adequately a function.	Count the number of users that ask for explanation and compare with the number of users that use the app without further explanation	Conduct user test and observe user behaviour.
Reliability	Availability	RA1	How available is the system for use during the specified period of time?	Test system in production for a specified period of time performing all user operations. Measure the time that the system is unavailable.	Conduct system test of the Integrated Solution (Apache Jmeter)
Portability	Installability and Adaptability	PI1	Can users successfully install the software in the operation environment?	Count number of installations successfully performed and compare with the number of attempts. A successfully installation means that the system was installed correctly and that all functions are working as expected or with minor problems.	Conduct installation and functional test using a representative set of mobile devices

Table 4.2: RESCUER quality in use metrics

Quality Attributes	**Quality Sub Attributes**	**Metric**	**Purpose of the metric**	**Method of application**	**Artefact**
Effectiveness		EF1	What proportion of the tasks are adequately completed by users?	Count total number of tasks adequately completed by users and compare with total number of tasks performed by users	Conduct user test
Efficiency		EF2	How long does it take to complete a task?	Count the number of users that think fast to perform a specific task a task	Conduct user test and interview user with questionnaire
Freedom from Risk		FR1	What is the incidence of hazard to people affected by use of the system?	Count number of people that feel to run further risks due to the use of the app and compare with the total number of people that used the app	Usage statistics

continued on next page ...

Quality Attributes	**Quality Sub Attributes**	**Metric**	**Purpose of the metric**	**Method of application**	**Artefact**
Satisfaction	Usefulness	SU1	How useful is the system for the user?	Measure the score assigned by users to usefulness criteria defined in the applied questionnaire	ISO Quest. Usability 9241-10 or Atrakdiff tool [5]
	Trust	ST1	How Trust is the system to the user?	Measure the score assigned by users to trust criteria defined in the applied questionnaire	ISO Questionnaire Usability 9241-10 or Atrakdiff tool [5]
Context Coverage	Context Completeness	CC1	How the system can be used with effectiveness, efficiency, freedom from risk, and satisfaction in all the specified contexts of use?	Compare the results of the Quality in Use metrics in different context of use	Evaluation report of the quality attributes: Effectiveness, Efficiency, Freedom from Risk, and Satisfaction

For each metric, acceptance criteria was defined used to decide whether the metric results were satisfactory or not, considering expected result in the final of project. These criteria were defined through detailed analysis of RESCUER quality requirements and refined by requirement team, task leaders, and project coordinators. Some of these acceptance criteria are shown in the next section together with the design of RESCUER SoS evaluation.

In addition, it is very important to highlight that specific input data is needed for an adequate application of these metrics. Input data can be obtained by using questionnaires, checklists, experiments, observations, simulations, etc. For each metric, methods of application, evaluation strategies, and sources of data that could be used in the measurement were established. This information was properly detailed in System Evaluation Plans created to guide evaluations of constituent systems and SoS as independent system.

4.4 Activity 3: Design of the SoS Evaluation

The design of RESCEUR SoS evaluation was developed to guide the evaluation of all constituent systems and SoS, during three iterations of the project. It also composes the acceptance criteria defined in last activity to better manage and control the quality evolution of the RESCUER SoS. This also allowed us to identify and, therefore, react in a straightforward manner, to problems that could influence the overall quality of RESCUER SoS. These acceptance criteria were defined considering an increased level of rigor, since metric results should improve in the course of the iterations to achieve the quality requirements expected to the final of the project.

Table 4.3[3] presents the design of SoS evaluation for the metrics UL1, UL2, and PT1, regarding the three project iterations, besides the increasing level of rigor for each acceptance criteria. As the application of metric PT1 in RESCUER SoS is dependent of all constituent systems working in integrated way, it was schedule to be applied to SoS only in the third project iteration.

Table 4.3: RESCUER evaluation design

Quality Attribute	**Quality Sub Attribute**	**Metric**	**First Iteration**		**Second Iteration**		**Third Iteration**	
			System	**Acceptance Criteria**	**System**	**Acceptance Criteria**	**System**	**Acceptance Criteria**
Usability	Learnability	UL1	MCS	60% of the users should adequately use the app without demonstration	MCS	65% of the users should adequately use the app without demonstration	MCS	70% of the users should adequately use the app without demonstration
		UL2	MCS	70% of the users should adequately use the app with demonstration	MCS	75% of the users should adequately use the app with demonstration	MCS	80% of the users should adequately use the app with demonstration
					ERTK	50% of the users should adequately use the app with demonstration	ERTK	75% of the users should adequately use the app with demonstration
Performance Efficiency	Time Behavior	PT1					SoS	The reports must not spend more than 3 min to be showed in ERTK System

All information produced until that moment were put in a document called SoS Evaluation Plan, mainly containing the SoS evaluation goals, SoS quality model, set of metrics, acceptance criteria, and SoS evaluation design. In general, this document was used as base to all expected evaluations during RESCUER project and, therefore, it was constantly adapted and detailed to better support the expected evaluations, mainly when the evaluation context became more clear for the evaluation team.

4.5 Activity 4: Execution of the SoS Evaluation

This section presents the execution of evaluation of MCS conducted in first project iteration, ERTK conducted in second project iteration, and RESCUER SoS conducted in third project iteration, considering the SoS Evaluation Plan previously established. The main idea is to present part of our experience evaluating this kind of systems and show evidences of the utility and applicability of the proposed SoS evaluation model in a real evaluation context. On the order hand, in Section 4.6 - Conclusion of the SoS Evaluation, results of all evaluation conducted in the context of RESCUER project are considered.

Therefore, it is important to point out exactly our participation in the planning, conduction, and report of these evaluations. During the RESCUER project, we participated actively on the planning, execution, and report of MCS evaluation and RESCUER SoS as a independent system. In very less intensity, we participated on the planning

[3]The full version of this table can be found in: https://goo.gl/F6qQ1K

of ERTK evaluation, as it was lead by others constituent organizations. However, all evaluations conducted during the RESCUER project used as basis the SoS Evaluation Plan established in the context of this dissertation. Details of each evaluation are described in internal project deliverables. In particular, in this chapter, these evaluations are presented in a summarized way to show the applicability of the SoS Evaluation Model proposed in this Master's project. Several others evaluations were conducted during this project, as defined in the SoS Evaluation Plan. In total, the RESCUER SoS and its constituent systems were evaluated 20 times, in different countries, context, and using different evaluation strategies. Section 4.6 presents an overview of all these evaluations.

Following the SoS Evaluation Model, each evaluation presented here is divided in three sub activities: (i) 4.1 Design of the System Evaluation; (ii) 4.2 Execution of the System Evaluation; and (iii) 4.3 Conclusion of the System Evaluation.

4.5.1 Evaluation of Mobile Crowdsourcing Solution

MCS is one of the main systems that composes the RESCUER SoS. It implements suitable context-sensitive mechanisms for eyewitnesses and operational forces carrying mobile devices to provide C&CC with information about emergency situations. Emergency situations trigger very basic human instinctive behavior, making people overwhelmed and confused. Therefore, it is important to deeply understand how these situations affect human behavior and how this can impact the MCS utilization.

This evaluation aimed to assess quality attributes, specifically: usability (user interface aesthetics, and learnability); satisfaction (usefulness, and trust); and effectiveness. These quality attributes were established to MCS considering the first project iteration. For each quality attribute, a set of corresponding metrics was pickup to measure the presence of such quality attributes on the MCS application. A summary of quality metrics considered in this evaluation are presented in Table 4.4. Through this evaluation, it was possible to verify if that current version of system achieved its quality requirements, and which improvements should be implemented in next system version.

Before starting to detail the design, execution, and conclusion of this evaluation, some important information about this constituent system is provided in next section.

4.5.1.1 Mobile Crowdsourcing Solution

The MCS supports communication of eyewitnesses and official first responders (e.g., police, fire fighter, etc.) with the C&CC. Eyewitnesses and first responders are equipped with MCS for the following purposes (Villela et al., 2013):

Table 4.4: MCS quality metrics

Quality Attributes	Quality Sub Attributes	Metric	Purpose of the metric	Method of application
Usability	**User interface aesthetics**	UU1	How attractive is the interface to the user?	Measure the score assigned by users to defined criteria in the applied questionnaire.
	Learnability	UL1	What proportion of user can operate successfully a function without a demonstration or tutorial?	Count the number of users that adequately operated the functions without a demonstration and compare with the total number of users in the evaluation who did not have a demonstration or tutorial.
		UL2	What proportion of user can operate successfully a function after a demonstration or tutorial?	Count the number of users that adequately operated the functions after the demonstration and compare with the total number of users in the evaluation who had a demonstration or tutorial.
Effectiveness		EF1	What proportion of the tasks are adequately completed by users?	Count total number of tasks adequately completed by users and compare with total number of tasks performed by users.
Satisfaction	**Usefulness**	SU1	How useful is the system for the user?	Measure the score assigned by users to usefulness criteria defined in the applied questionnaire.
	Trust	ST1	How Trust is the system to the user?	Measure the score assigned by users to trust criteria defined in the applied questionnaire

- Eyewitnesses use MCS on their mobile devices to provide multimedia information about an incident that has occurred. The goal is to benefit as much as possible from information that can be provided by mobile devices without any explicit action of their users, but taking into consideration the user's privacy; and

- First responders focus on rescuing victims, providing medical care, and dealing with hazards, so their profile for iteration with the command centre is very similar to the eyewitnesses' profile. They mainly use mobile devices, such as smart phones and wearable devices, equipped with MCS to keep C&CC informed about the evolution of the situation.

Based on that, two types of information can be gathered from people carrying mobile devices in the place of an emergency situation: (i) information that can be extracted from mobile devices without user interaction with those devices, e.g., GPS position, movement speed, movement trails, and number of devices at a specific location; and (ii) information provided by users through an explicit interaction with their mobile device, e.g., videos of the incident, text message with the incident description, and photos of damages.

Figure 4.3 presents the user interface of MCS, which was used in this evaluation. In short, users start the report of an incident by notifying the C&CC. Users can select the incident type, such as explosion, fire, human crush, among others (on the first screen). By pressing one of the options, the reporting process is immediately triggered and sensor based information is sent to the server. After this, the user can continue the interaction and send a standard report or call back the notification, in the case of a false alarm. In a standard report (on the second screen), the user is confronted with his/her own position (automatically detected) represented in terms of a pin displayed on the map and the incident position, which can be redefined by holding and dragging the point of the incident on the map. In addition, this screen offers the possibility of specifying the severity of the incident, if there are injured persons, and possibility of taking photos/videos of the incident. By navigating in the section "Describe the fire", the user get a new screen (on the third screen) with several attributes that are relevant for the incident classification.

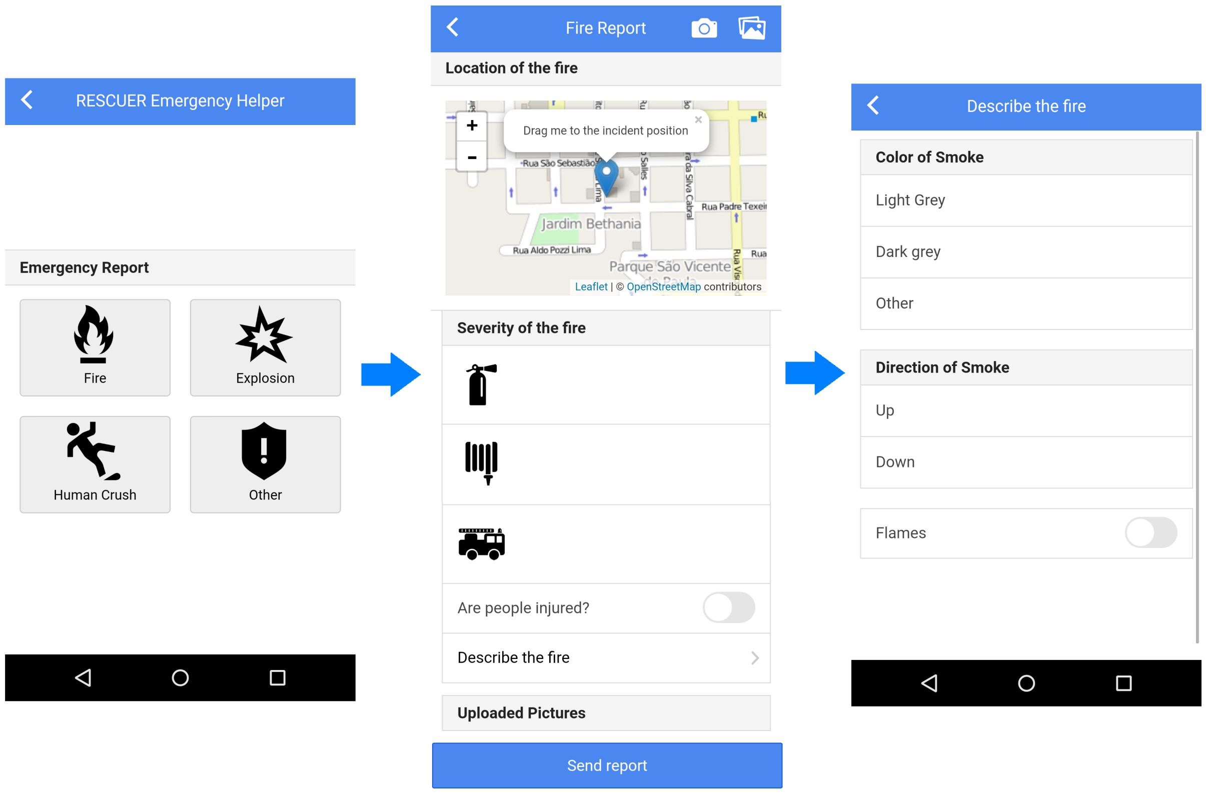


Figure 4.3: MCS user interface (RESCUER, 2015)

4.5.1.2 Activity 4.1: Design of the System Evaluation

To evaluate the MCS, a System Evaluation Plan was prepared. This plan aims to define strategies, guidelines, artifacts, scenarios, and participants that are considered to obtain information for the quality assessment. In short, to obtain the needed data to assessment of set of quality attributes defined to this evaluation, two complementary strategies were considered: (i) evaluation of MCS in use; and (ii) interviewee opinion and characterization through a survey. These strategies are described in more detail bellow.

(i) Evaluation of MCS in Use

In this strategy, the evaluation was performed considering the use of MCS by potential eyewitnesses in the scenarios of large events and industrial parks (these scenarios will be detailed in Section 4.5.1.3). In summary, potential users were asked to perform a predefined set of tasks in the context of an emergency situation such as a fire in a stadium. In the meanwhile, an evaluation team guided the tasks execution and observed the participants behaviour. The tasks asked for participants to perform were:

- T1. Report that you see an incident;
- T2. Inform that the incident is in the other side of yours;
- T3. Inform that you see injured people;
- T4. Describe the properties of the incident;
- T5. Inform the severity of the incident; and
- T6. Take a photo of the incident.

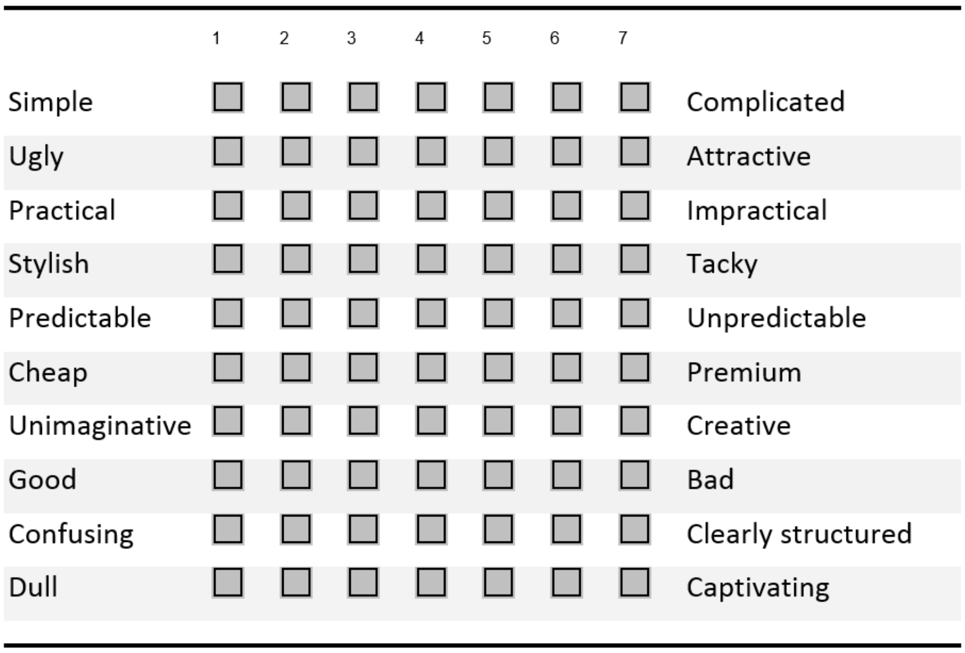
Each evaluation team was composed by two people, one moderator, and one observer, each one with the following responsibilities: Moderator was responsible for addressing participants, presenting the application and an emergency scenario, supporting participants during the test, and applying the survey in the final of the evaluation; Observer was responsible to observe if the users performed each task in an expected way filling up the observation sheet and collecting evaluation cards used in the survey. In addition, in each evaluation, one person was responsible for supporting and supervising the activities performed by evaluation teams.

For each task set, users were randomly divided in two groups: (i) users that performed the tasks without previous demonstration of MCS; and (ii) users that performed the tasks after demonstration. This division in groups was very important, as it allowed us to evaluate the learnability of MCS and to identify which aspects and characteristics of the application can influence its usability. It was expected that the difference between the results for these groups would be minimal, what would indicate that the application is usable enough for the users to complete all tasks without difficulties, doubts or questions even in an emergency situation.

(ii) MCS Survey

After using the application, users were asked to answer questions in a survey to provide their opinion about the usability of the application. They also asked questions regarding general acceptance of the application and their personal characteristics.

The questionnaire applied to obtain feedback from users, regarding the usability of the application was defined based on the AttrakDiff questionnaire (Väätäjä et al., 2009). This is an established evaluation tool that addresses evaluations of user experience and has already been used for evaluating mobile systems (Väätäjä et al., 2009). As it can be seen in Figure 4.4, this questionnaire consists of pairs of contrasting attributes that may be applied to the application, such as Simple and Complicated, Ugly and Attractive, Confusing and Clearly structured, among others. In this questionnaire, squares between attributes represent gradations between the opposites. The user can express his/her agreement with the attributes by ticking the square that most closely reflects his/her impression.



	1	2	3	4	5	6	7	
Simple	☐	☐	☐	☐	☐	☐	☐	Complicated
Ugly	☐	☐	☐	☐	☐	☐	☐	Attractive
Practical	☐	☐	☐	☐	☐	☐	☐	Impractical
Stylish	☐	☐	☐	☐	☐	☐	☐	Tacky
Predictable	☐	☐	☐	☐	☐	☐	☐	Unpredictable
Cheap	☐	☐	☐	☐	☐	☐	☐	Premium
Unimaginative	☐	☐	☐	☐	☐	☐	☐	Creative
Good	☐	☐	☐	☐	☐	☐	☐	Bad
Confusing	☐	☐	☐	☐	☐	☐	☐	Clearly structured
Dull	☐	☐	☐	☐	☐	☐	☐	Captivating

Figure 4.4: AttrakDiff based questionnaire (RESCUER, 2015)

To identify the acceptance of the application, following questions were used in the questionnaire: (i) Would you use this application to help workforces if an emergency situation like this occurs during a large event? (ii) Would you use this application to safe yourself if an emergency situation like this occurs during a large event? In addition, users were asked to answer some personal information, such as (i) Gender; (ii) Age; (iii) If they own a smartphone; and (iv) If they have experience with emergency situation. Through this questionnaire, it was also possible to obtain specific recommendation of the users to become the interface more intuitive as possible.

4.5.1.3 Activity 4.2: Execution of the System Evaluation

Considering the scenario of large events, this evaluation was performed in Salvador - Brazil, São Carlos - Brazil, and Kaiserslautern - Germany. In particular, the evaluation happened during FIFA World Cup 2014, which is one of the biggest sport events of the world and was used as main scenario of MCS evaluation. In this scenario, 50 people participated in the evaluation conducted in Kaiserslautern - Germany, 35 in Salvador - Brazil, and 27 in São Carlos - Brazil, totalizing, 112 participants, 55 without demonstration and 57 with demonstration of MCS.

Considering the scenario of industrial parks, the evaluation was performed in Camaçari Industrial Complex[4], which is the largest integrated industrial complex in the Southern Hemisphere. It is comprised of over 90 chemical and petrochemical companies, besides other production facilities, such as cellulose, copper metallurgy, textiles, automobiles, beverages, and services. In this scenario, a total of 60 people participated in the evaluation conducted in Camaçari - Brazil, 28 without demonstration and 32 with demonstration.

4.5.1.4 Activity 4.3: Conclusion of the System Evaluation

Results presented in this section were grouped according to tasks performed by each participant with and without demonstration of the application. For each task defined in the planning of this evaluation, the number of users that performed it successfully was measured. A successfully performed task means that no question was asked and the participant behavior was as expected. In addition, the usability of application can be derived from the slight difference between results of those who received and did not receive a demonstration. This means that, as lower the influence of demonstration in the comprehension of application and effectiveness of users, the level of usability and learnability of the application will be greater.

Table 4.5 shows a summary of the successfully performed tasks regarding the kind of demonstration and evaluation scenario. As it can be observed, tasks "T2. Inform that the incident is in the other side of yours", "T3. Inform that you see injured people", and "T6. Take a photo of the incident" had a lower level of effectiveness. Beside this, the difference of effectiveness regarding users that received and not received demonstration was very significant. This allow us to identify key points of application that impacted the usability of the application in these tasks.

Table 4.5: Percentage of tasks successfully performed

Tasks	Large scale events		Industrial Park	
	With Demo	Without Demo	With Demo	Without Demo
T1. Report that you see an incident	95%	80%	88%	82%
T2. Inform that the incident is in the other side of yours	59%	38%	86%	61%
T3. Inform that you see injured people	80%	67%	80%	76%
T4. Describe the properties of the incident	82%	72%	80%	60%
T5. Inform the severity of the incident	81%	58%	91%	89%
T6. Take a photo of the incident	62%	49%	72%	61%

The feedback of the participants regarding the usability of the application, as early mentioned, was captured through the use of an AttrakDiff questionnaire. In general, the feedback of the participants was positive. On average, the score obtained was 6 of a

[4]http://www.coficpolo.com.br

maximum of 7. In addition, the application was well received by the general public. More than 85% of participants said they would use MCS to save themselves, which indicates a clear acceptance. When asked if they would use the application to help operational forces, more than 90% responded positively. With these results, it is possible to conclude that most people would probably use this application in an emergency situation.

Using the information obtained from evaluation of MCS (in use and through the survey), the set of metrics presented in Table 4.4 was calculated. After this, results were compared to acceptance criteria established for that current project iteration. Table 4.6 presents the MCS evaluation results regarding quality attributes and scenario where evaluation was performed (i.e., large scale events or industrial park). To better understanding how the metric results were obtained, some clarifications are provided.

Table 4.6: MCS evaluation results

Quality Attribute	Quality Sub Attribute	Metric	Acceptance criteria	large-Scale Events	Industrial Park	Total Measure	Total Result
Usability	**User Interface Aesthetics**	UU1	0.7	0.84	0.87	0.86	yes
	Learnability	UL1	0.6	0.57	0.70	0.64	yes
		UL2	0.7	0.73	0.80	0.77	yes
Effectiveness		EF1	0.55	0.66	0.76	0.71	yes
Satisfaction	**Usefulness,**	SU1	0.6	0.94	0.97	0.96	yes
	Trust	ST1	0.6	0.87	0.93	0.90	yes

The metric UU1 was calculated from the participants' feedback about the usability of application. In particular, the average score for all aspects defined in AttrakDiff questionnaire was used as input for the measurement of this metric. In this sense, 0.7 is the acceptance criteria defined for this metric; 0.84 and 0.87 are the average score obtained in AttrakDiff questionnaire for large-scale events and industrial parks context, respectively; 0.86 is the average of results obtained in both contexts; and finally, column "Total Results" shows that the total measure was sufficient, when compared to the acceptance criteria. Metrics UL1 and UL2 were calculated from the number of all tasks performed by participants with and without demonstration of MCS, respectively. Metric EF1 was calculated from the number of tasks successfully performed by participants. Metrics SU1 and ST1, regarding the user satisfaction, were measured considering answers obtained in the application acceptance evaluation. Specifically, answers to question "Would you use this application to safe yourself?" were used to measure the degree to which a user has confidence in the application, whereas answers to question "Would you use this application to help operational forces?" were used to measure the degree to usefulness of the application perceived by users.

As presented in Table 4.6, all results were satisfactory considering our expectations for that evaluation iteration. This means that, taking into consideration the average evaluation results, the metric values are higher than the values of the acceptance criteria.

Despite this, it was observed the existence of room for improvement to achieve a higher quality for the next evaluation iteration and to make the application as intuitive as possible. All improvement points identified were carefully considered and analyzed to develop a better version of MCS.

4.5.2 Evaluation of Emergence Response Toolkit

ERTK and DAS are two of four main constituents of RESCUER SoS. ERTK is the interface between RESCUER SoS and dispatchers at C&CC; it provides visualization of all data analysed by DAS in a user-friendly application. On the other hand, DAS provides accurate information about the incident, as reported by the crowd, in the minimum possible time. Therefore, this section describes results of ERTK evaluation that was carried out in different C&CC in Brazil and Austria.

This evaluation aimed to assess quality attributes, more specifically, usability (user interface aesthetics, learnability, and operability); satisfaction (usefulness); efficiency; and effectiveness. These quality attributes were established by the SoS Evaluation Plan considering the second project iteration. For each quality attribute, a set of corresponding metrics was pickup to measure their presence on ERTK. A summary of the quality metrics considered in this evaluation is presented in Table 4.7. Through this evaluation, it was possible to verify if that version of ERTK meets its quality requirements, and which improvements should be implemented in next ERTK version.

Table 4.7: ERTK quality metrics

Quality Attribute	**Quality Sub Attribute**	**Metric**	**Purpose of the metric**	**Method of application**
Usability	Appropriateness Recognisability	UA1	What proportion of the product functions will the user be able to understand correctly?	Conduct user test and interview user with questionnaires. Count the number of user interface functions where purposes are easily understood by the user and compare with the number of functions available for user.
	Learnability	UL3	What proportion of users have clearly understood the purpose of the available functions?	Count the number of users that have clearly understood the purpose of the available functions and compare with the total number of users in the evaluation.
	Operability	UO2	What proportion of users think the system easy to use and navigate?	Count the number of users that think the system easy to use and compare with the total number of users in the evaluation.
Effectiveness		EF1	What proportion of the tasks are adequately completed by users?	Count total number of tasks adequately completed by users and compare with total number of tasks performed by user
Efficiency		EF2	How long does it take to complete a task?	Measure the average time to perform a task
Satisfaction	Usefulness	SU1	How useful is the system for the user?	Measure the score assigned by users to usefulness criteria defined in the applied questionnaire

Before to detail the design, execution, and conclusion of this evaluation, some important information about this constituent system is provided in next section.

4.5.2.1 Emergence Response Toolkit

This section describes briefly ERTK and also the DAS, which is responsible to fed ERTK. DAS provides a multimedia description of the emergency situation, as complete as possible. This constituent system receives text and multimedia data from MCS used by the crowd, and then, the output of DAS is visualized and managed in the ERTK (RESCUER, 2016a).Three main data analysis are performed by DAS depending on the type of data received (RESCUER, 2016a):

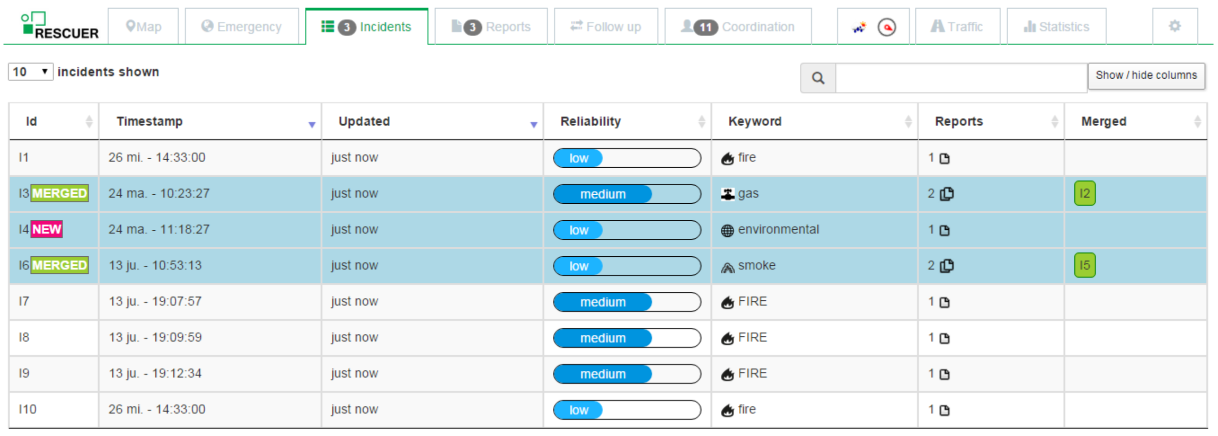
- Video Analysis: Video analysis component analyzes video and sensor data to provide high level measures (e.g., crowd density, risk of congestion) about the crowd in the scenario and detects fire and/or smoke in the scene;
- Image Analysis: Image analysis component focuses on detecting and characterizing fire and/or smoke in every image the system receives. This component is composed by a classifier based on colour detection methods and similarity-queries; and
- Text Analysis: Text analysis component extracts relevant emergency-related information by the text data (e.g., SMS, incident reports). The main purpose of this component is to extract most important parts of the information: what is happening, where, and who are involved.

The main goal of ERTK is to provide necessary and sufficient information at the right time to C&CC staff, by means of an intuitive and concise user-interface that facilitates the decision taking process when dealing with an emergency (RESCUER, 2016a). The following list presents all concepts in which the features of ERTK are built on (RESCUER, 2016a):

- Emergency Map: Features that enable showing the location of the incidents, drawing and text annotations, crowd density, and behaviour visualisation;
- Emergency Dashboard: Features that show statistics related to incidents, traffic, and weather information;
- Emergency Browser: Features that allow to explore information about the emergency situation; and
- Follow-up Interaction/Guidance Messages: Features to send messages based on location and/or specific MCS user profiles.

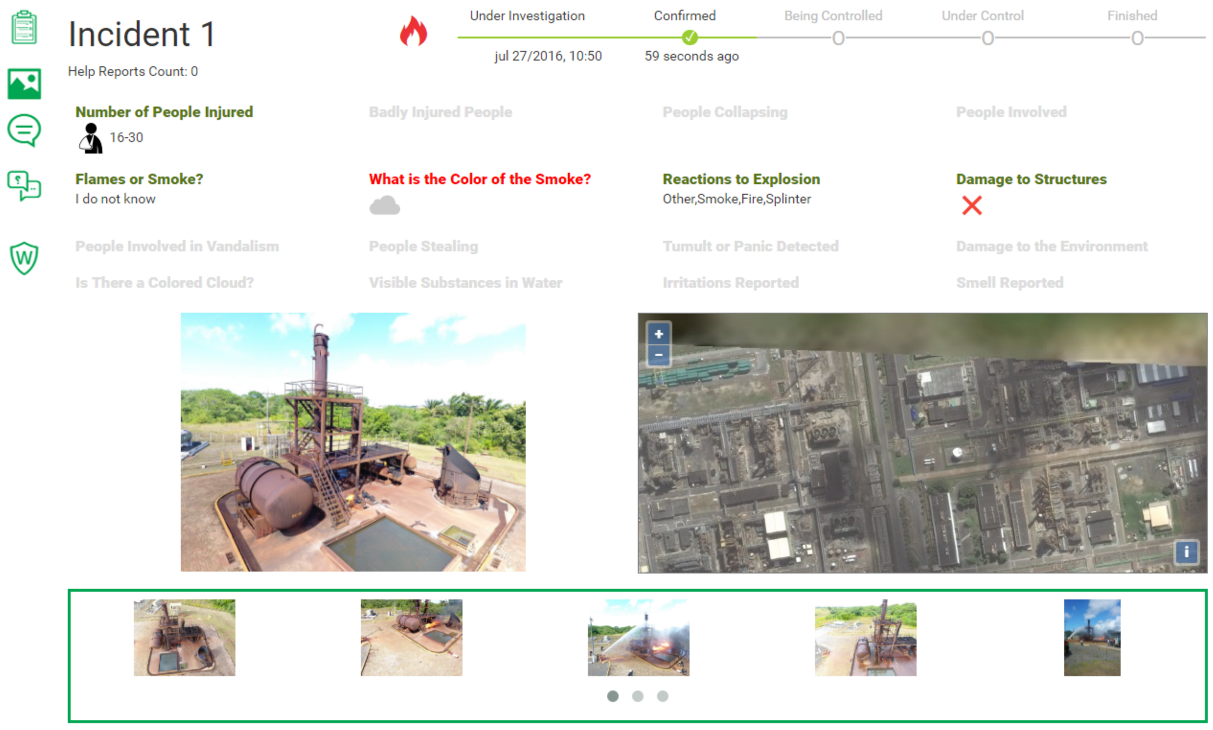
Basically, an emergency in the ERTK is defined by reports sent from the MCS. ERTK processes these reports and identifies the incidents reported. The incident browser (see

Figure 4.5) shows the list of incidents with key information obtained from reports of each incident. The incident detail view (see Figure 4.6) contains several visualization components that may be used to get more information about an incident, and also to see details about each report from the incident selected. In map view (see Figure 4.7), the location and movement of all MCS users inside of monitoring zone are presented, showing an overview of emergency situation. Besides these features, ERTK is composed of several others. For more details about all ERTK features, see (RESCUER, 2016a).



Id	Timestamp	Updated	Reliability	Keyword	Reports	Merged
I1	26 mi. - 14:33:00	just now	low	fire	1	
I3 MERGED	24 ma. - 10:23:27	just now	medium	gas	2	I2
I4 NEW	24 ma. - 11:18:27	just now	low	environmental	1	
I6 MERGED	13 ju. - 10:53:13	just now	low	smoke	2	I5
I7	13 ju. - 19:07:57	just now	medium	FIRE	1	
I8	13 ju. - 19:09:59	just now	medium	FIRE	1	
I9	13 ju. - 19:12:34	just now	medium	FIRE	1	
I10	26 mi. - 14:33:00	just now	low	fire	1	

Figure 4.5: ERTK incident browser (RESCUER, 2016a)



Incident 1

Help Reports Count: 0

Under Investigation — jul 27/2016, 10:50
Confirmed — 59 seconds ago
Being Controlled
Under Control
Finished

Number of People Injured
16-30

Badly Injured People

People Collapsing

People Involved

Flames or Smoke?
I do not know

What is the Color of the Smoke?

Reactions to Explosion
Other,Smoke,Fire,Splinter

Damage to Structures
✗

People Involved in Vandalism

People Stealing

Tumult or Panic Detected

Damage to the Environment

Is There a Colored Cloud?

Visible Substances in Water

Irritations Reported

Smell Reported

Figure 4.6: ERTK incident detail view (RESCUER, 2016a)

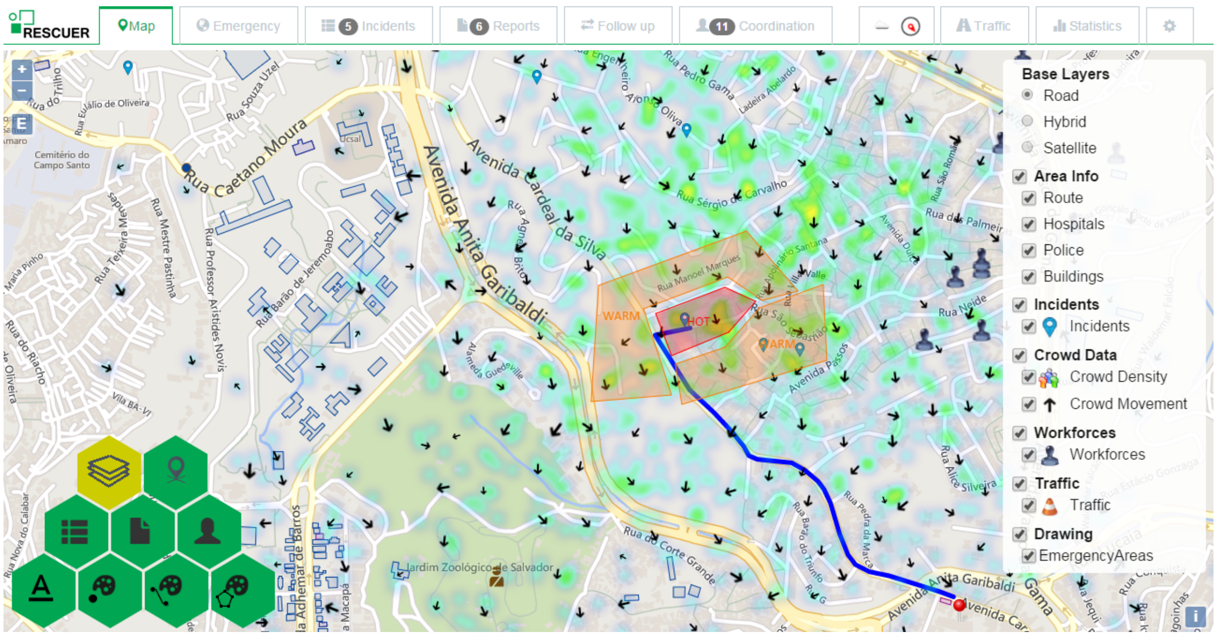


Figure 4.7: ERTK map view (RESCUER, 2016a)

4.5.2.2 Activity 4.1: Design of the System Evaluation

To guide the evaluation of the ERTK, a System Evaluation Plan was prepared. This plan aimed to define strategies, guidelines, artifacts, scenarios, and participants that would be considered to obtain information for the quality assessment. To obtain needed data to evaluate the set of quality attributes defined to this evaluation, the evaluation was conducted with emergency management specialists.

In this evaluation, it was considered an emergency scenario where an explosion happened during a large-scale event and people in the crowd started to send reports through the MCS. During this scenario, volunteers played the role of a C&CC staff and were asked to perform a set of tasks, as presented bellow:

- T1. Identify the type of incident that happened and where it happened;
- T2. Find the screen with information about the incident;
- T3. List all available information about the incident;
- T4. Change the incident status to confirmed;
- T5. In case you identify, list changes in the information regarding the incident;
- T6. Identify the color of the smoke;
- T7. Again, in case you identify, list changes in the information regarding the incident; and

- T8. Colors indicate the user's profile that sent an information (e.g., civilian, supporting force). According to you feelings, associate a color to a profile.

During execution of these tasks, two members of evaluation team sent to ERTK predetermined incident reports at a specific time using the MCS, and considering the established scenario. This made possible to compare the information provided in the incident reports and that one identified by participant in each task. The participants were not trained; however, during the tasks execution, they could ask to any support or even to skip a task. All the time, participants behaviour was observed by a member of evaluation team that was aware to the expected reactions for each task, and took notes about what the participant did, besides, this member was also responsible to give to participants each evaluation task in the right time.

After conducting the evaluation, participants were asked to answer an evaluation questionnaire and a characterization form. Basically, the evaluation questionnaire was composed of a set of question, such as: (i) How easy it was to find the type of incident and its location?; (ii) How useful do you consider the new incident alert?; (iii) In your opinion, is it easy to navigate through the system?; (iv) How fast do you think you were in finding information about the incident?; (v) How clear are the terminologies used from the current incident status?; (vi) How easy it was to notice that a new information arrived or that an old information changed?; (vii) Have you managed to understand the meaning of the icons in the incident information screen?; (viii) In you opinion, how difficult it was to understand the trust level of each information?; and (ix) How useful do you think this system is to help a command and control center in obtaining information about an incident, and aiding in decision making? The answers of all these questions were used as input for the metrics and, consequently, the measurement of ERTK quality attributes.

4.5.2.3 Activity 4.2: Execution of the System Evaluation

This evaluation was conducted in two different places, firstly, at CEIC - Centro Integrado de Comando[5] (in English, Command Integrated Center) in Porto Alegre, Brazil. It is the intelligence center of the city, guaranteeing security of citizen through the integration of video-monitoring, operational planning of large events, climate monitoring, and emergency response. This evaluation was also performed during CIDEM - Congresso Internacional de Desastres em Massa[6] (in English, International Congress on Mass Disasters) in Salvador, Brazil. This event was attended by members of local, regional, national, and international organizations that act in disaster situations, such as civil defence, municipal guard, firefighters, military police including the special operations police battalion

[5]http://www2.portoalegre.rs.gov.br/ceic/
[6]http://www.cidem2016.com.br/

(BOPE), civil police, technical police, federal police, army, navy, air force, interpol and universities. This occurred from 10 to 12 June 2016 in both Hotel Fiesta and Arena Fonte Nova, with the theme: "Safety for large events - A global warning".

In both opportunities, emergency management specialists and potential ERTK users were approached and asked to participate in this evaluation. For all participants, it was provided an overview of RESCUER purpose and an explanation about the goals of the evaluation, besides the role of the participant in the evaluation.

4.5.2.4 Activity 4.3: Conclusion the System Evaluation

In total, 15 volunteers participated in this evaluation, six from CIDEM and nine from CEIC. In average, they had 13 years of experience in emergency management. 80% of them had experience with web systems, and 53% of them had some experience with emergency management systems.

About the execution of tasks, participants spend more time trying to accomplish the task "T2 - Find the screen with information about the incident". Besides that, this task was skipped two times by participants during evaluation. Overall, participants did not spend more than one minute to conclude each task, considering the time spend with some discussion about participants feedback, and feeling in each task given.

As presented in Figure 4.8, evaluation results were positive, with most of participants successfully performing all tasks. Besides that, participants with and without experience in emergency systems finished the tasks in almost same time, which raises evidences about the usability of ERTK.

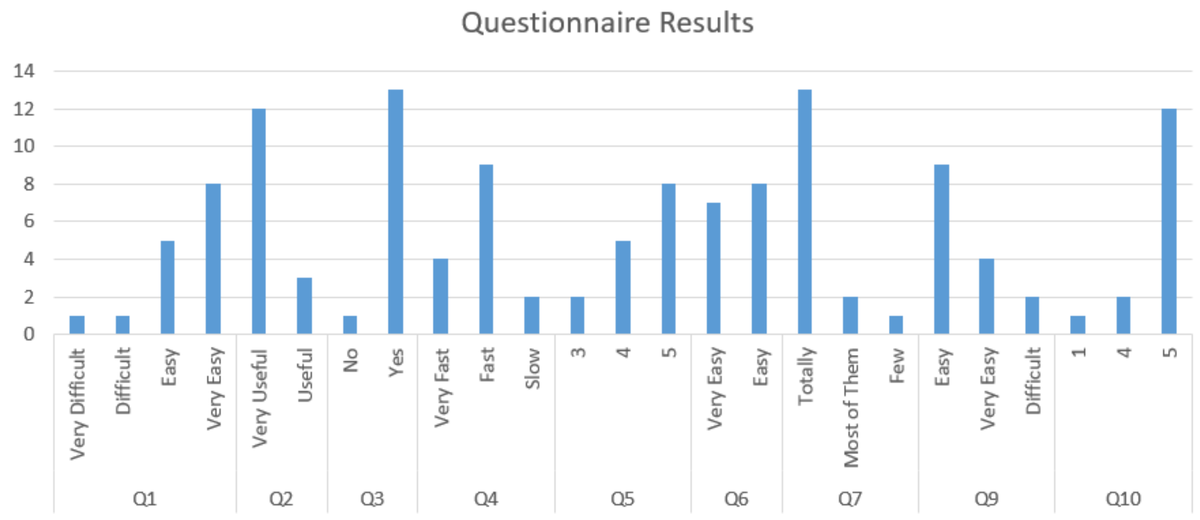


Figure 4.8: ERTK questionnaire results

Using information obtained from the evaluation of ERTK, the set of metrics presented in Table 4.7, defined for the measurement of quality attributes addressed to ERTK, was calculated. About results obtained through the application of questionnaire, Table 4.8 summarizes the relation between questions used as source of data and quality attributes. In the case of more than one question contributing to a quality attribute, the final measure is the average of the results of each question. On the other hand, the effectiveness is measured by the number of tasks successfully performed by the participants.

Table 4.8: Relation between questions and metrics

Question	**Quality Attribute**	**Metric**
Q1. How easy it was to find the type of incident and its location?	Usability (Appropriateness Recognisability)	UA1
Q2. How useful do you consider the new incident alert?	Satisfaction (Usefulness)	SU1
Q3. In your opinion, it is easy to navigate through the system?	Usability (Operability)	UO2
Q4. How fast do you think you were at finding the information about the incident?	Efficiency	EF2
Q5. How clear are the terminologies used from the current incident status?	Usability (Appropriateness Recognisability, Learnability)	UA1, UL3
Q6. How easy it was to notice that a new information arrived or that an old information changed?	Usability (Appropriateness Recognisability)	UA1
Q7. Have you managed to understand the meaning of the icons in the incident information screen?	Usability (Appropriateness Recognisability, Learnability)	UL3, UA1
Q8. In you opinion, who difficult it was to understand the trust level of each information?	Usability (Appropriateness Recognisability, Learnability)	UL3, UA1
Q9. How useful do you think this system is to help a command and control center in obtaining information about an incident, and aiding in decision making?	Satisfaction (Usefulness)	SU1

After this, results were compared to acceptance criteria established in the SoS Evaluation Plan. Table 4.9 presents results of quantitative evaluation of ERTK.

Table 4.9: ERTK evaluation results

Quality Attribute	**Quality Sub Attribute**	**Metric**	**Accept. Criteria**	**Measurement**	**Result**
Usability	Appropriateness Recognisability	UA1	0.8	0.94	Yes
	Learnability	UL3	0.8	0.94	Yes
	Operability	UO2	0.8	0.93	Yes
Effectiveness		EF1	0.8	0.98	Yes
Efficiency		EF2	0.8	0.87	Yes
Satisfaction	Usefulness	SU1	0.8	0.97	Yes

In short, all results were satisfactory considering our expectations for this project iteration. This means that, taking into consideration the average evaluation results, the metric values were higher than the values of acceptance criteria. Despite this, it was observed the existence of room for improvement that was carefully considered and analyzed to make the application as intuitive as possible.

4.5.3 Evaluation of RESCUER SoS as Independent System

Since the individual evaluation of each main RESCUER constituents is not able to guarantee the quality of whole system, the evaluation of RESCUER SoS as an independent system must be conducted.

This evaluation aimed to assess the following quality attributes: performance efficiency (resource utilization, capacity, and time behavior); reliability (availability); functional suitability (completeness); usability (appropriateness recognisability); freedom from risk; satisfaction (usefullness and trust). These quality attributes were established in the SoS Evaluation Plan, considering the third project iteration. For each quality attribute, a set of metrics was pickup to measure the presence of such quality attributes on the RESCUER SoS. Table 4.10 presents a summary of quality attributes and metrics considered in this evaluation.

Table 4.10: RESCUER SoS metrics

Quality Attribute	**Quality Sub Attribute**	**Metric**	**Purpose of the metric**	**Method of application**
Performance Efficiency	Resource Utilization	PR2	How adequate is the processor consumption?	Measure the processor consumption for a specific period of time and compare with the acceptable processor consumption
		PR3	How adequate is the memory consumption?	Measure the memory consumption for a specific period of time and compare with the acceptable memory consumption
	Capacity	PC1	What is the rate of incident reports that system can handle without problems?	Send the expected load of incident reports and measure the rate that the system can handle.
	Time Behaviour	PT1	What is the average time that a report sent from MCS take to be presented by ERT	Simulate an expected load of users, and measure the average time to reports be presented by ERT
Reliability	Availability	RA1	How available is the system for use during the specified period of time?	Test system in production for a specified period of time performing all user operations. Measure the time that the system is unavailabe
Functional Suitability	Completeness	FF1	How complete is the system from the user point of view?	Count the number of user that think the system complete and compare with the total number of users in the evaluation.
Usability	Appropriateness Recognisability	UA1	What proportion of the product functions will the user be able to understand correctly?	Count the number of user interface functions that are easily understood by the user and compare with the number of functions available for user
Freedon from Risk		FR1	What is the incidence of hazard to people affected by use of the system?	Count number of people that feel to run further risks due to the use of the app and compare with the total number of people that used the app
Satisfaction	Usefulness	SU1	How useful is the system for the user?	Measure the score assigned by users to usefulness criteria defined in the applied questionnaire
	Trust	ST1	How Trust is the system to the user?	Measure the score assigned by users to usefulness criteria defined in the applied questionnaire

4.5.3.1 Activity 4.1: Design of the System Evaluation

To guide this evaluation, a System Evaluation Plan was prepared. This plan aimed to define strategies, guidelines, artifacts, scenarios, and participants to obtain information for quality assessment. To obtain the needed data to application of the set of metrics, two evaluation strategies were conducted: (i) load test experiment to measure performance efficiency and reliability; and (ii) emergency simulation exercise to measure functional suitability, usability, freedom from risk, and satisfaction. Details about the design of these evaluation strategies are presented bellow.

Load Test Experiment

This load test experiment aimed to simulate a load of incident reports sent by eye-witness in a large-scale emergency situation. This experiment was supported by tool Apache Jmeter[7]. Apache Jmeter is an open source and a Java application, designed to test functional behaviour and to measure system performance. The goal was to obtain, in an efficient way, all input data to evaluate performance efficiency and reliability.

The load test planning intended to, simultaneously, simulate thousands of users of MCS application sending incident reports, while RESCUER infrastructure was monitored. Using Apache Jmeter, it was possible to analyze performance and to test the server behavior under a high concurrent load.

The load test was planned considering a simulation of a fire incident, divided in three phases with different load of reports. The total time of the incident was one hour, i.e., the idea was to simulate an incident that started and was combated in a period of one hour, when no more reports were sent. During this period, C&CC members could use all information provided by crowd to any needed decision making. The time and number of reports that were sent in each incident phase are described below.

- 1st phase: 3,000 reports sent in 10 minutes
- 2nd phase: 3,000 reports sent in 20 minutes
- 3rd phase: 3,000 reports sent in 30 minutes

An application was developed by one of constituent organizations to creates a JSON file with a set of 9000 reports describing a fire incident, and considering the following distribution: 50% of reports with only text; 45% with images; and 5% with videos. This JSON file is consumed by Jmeter that send the reports according the incident phases described above. Results of this load test allowed the identification and correction of potential bugs in the RESCUER SoS.

[7]http://jmeter.apache.org/

Emergency Simulation Exercise

A emergency simulation exercise was conducted in an still unopened tunnel near the town of Lambach, close to Linz, Austria. The tunnel has been built by Caverion Group[8], and is 912m long. A diagram of the tunnel can be seen in Figure 4.9.

The incident simulated was a car accident in the tunnel, with three vehicles involved. One of them was represented to be on fire, with smoke in the tunnel represented by cold smoke. During the incident, 13 people were involved: one person trapped in one of the vehicles, two people injured, but capable of walking, and 10 other people not injured and able to walk. This simulation also included a number of participants from emergency response entities, such as 30 firefighters, six polices, 10 rescue service members, besides more 20 people from tunnel administration and control centre personnel, road maintenance depot staff, state authorities personnel, company employees, and public authorities. Most of them had an active role in the emergency simulation exercise. The evaluation team was composed of eight people divided in five groups, as presented in Table 4.11, which describes the local, tasks to be performed, and number of participants for each evaluation team group. In summary, two groups were inside the Lambach tunnel supporting participants and sending incident reports when necessary. One person was responsible to take pictures and videos of all activities. Inside the C&CC, one person was monitoring the ERTK and one was operating the RESCUER news, which is a module of the system that supports the public communication of the incident status. In total, 86 people participated in this emergency simulation.

[8]http://www.caverion.com/

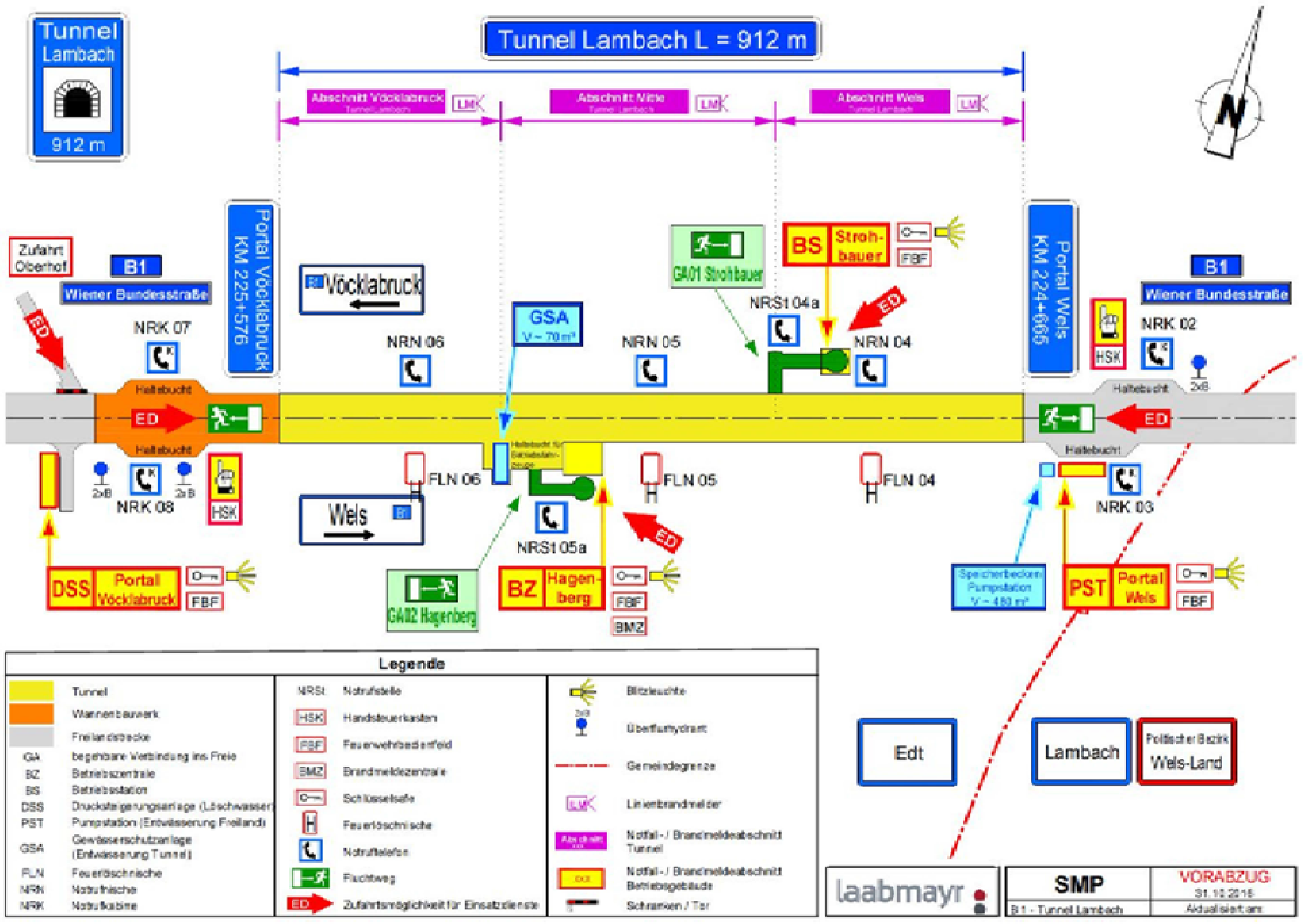


Figure 4.9: Diagram of the Lambach tunnel (RESCUER, 2016b)

Table 4.11: Evaluation team distribution (RESCUER, 2016b)

Team	**Local**	**Tasks**	**Participants**
Team 1	Tunnel	Support to participants, sending reports from Civilian or Supporting force profiles when necessary.	3 people
Team 2	Tunnel	Support to participants, sending reports from Workforce profiles when necessary.	2 people
Reporter	Tunnel	Record pictures and videos of the activities	1 person
Coordinator	C&CC Linz	Monitor the system	1 person
News person	C&CC Linz	Operate RESCUER News	1 person

In summary, this simulation exercise was divided in five phases:

- **Preparation:** During this phase, preparation activities were performed. This included contacting people who would participate in the evaluation, providing them with the necessary materials for the evaluation. Other activities included the technical deployment, setup and operation of the RESCUER system components, checking of communications and data coverage in the evaluation scenario, etc.;

- **Opening:** During this phase, the final preparation activities were carried out, including checking whether users had all materials and MCS application installed, if ERTK was functioning correctly, etc. Additionally, a script was rehearsed with all participants and RESCUER evaluation team to guide the use of the application through the evaluation;

- **Simulation start:** The incident started as scripted, with the three cars and the trapped person surrounded by smoke. In this moment, people capable of walking informed the emergency situation via both, the tunnel incident mechanisms and the MCS. After that, they proceeded to the designated tunnel emergency rescue rooms;

- **Emergency responses:** Having been alerted, firefighter brigades arrived at the tunnel and began the actions to control the fire. At the same time, the extractors of the tunnel were activated at maximum power to quickly dissipate the smoke. The smoke intensified several times during the exercise to emulate a worsening of the conditions of the accident. When the fire was getting under control, the firefighters proceeded to extract the trapped person from the car and moved her to a safe location for first aids. The ambulance was at the entrance of the tunnel, waiting for the fire brigades to grant them access to the tunnel and evacuate the injured people. All the time, incident reports were sent using the MCS application, providing additional information, photos of the incident, and incident status updates; and

- **Emergency under control/finished:** In this phase, the incident was considered to be under control, and the ambulance was allowed to get to the accident site, and evacuate the injured person. The corresponding incident updates were sent to C&CC.

After all evaluation phase to be concluded, a feedback was asked to participants via a questionnaire. This questionnaire was composed of eight questions: (i) Does RESCUER provide its information in a timely manner to adequately support decision making?; (ii) Can the use of RESCUER SoS reduce the use of human and material resources needed to manage an emergency situation?; (iii) Do you trust in RESCUER SoS to receive information from people in the incident place?; (iv) Do you trust that information presented by RESCUER SoS represents the real situation in the incident place?; (v) Do you think the use of RESCUER SoS in an emergency situation can expose people in the incident place to additional risks?; (vi) Would you actually use RESCUER SoS for emergency management?; (vii) Would you recommend the use of RESCUER SoS to other emergency response entities or public authorities?; and (viii) Is there any other activity related to emergency management that RESCUER should support?

The answers of all these questions were used as input for the metrics and, consequently, the measurement of quality attributes considered in the evaluation of RESCUER SoS.

4.5.3.2 Activity 4.2: Execution of the System Evaluation

The load test experiment was conducted in September 09, 2016 from 00:34am to 01:34am. The load of 9,000 reports was sent from Apache Jmeter to ERTK following the scenario defined in the System Evaluation Plan. During the experiment, RESCUER SoS was monitored regarding performance and reliability aspects.

The emergency simulation exercise was conducted in November 16, 2016 starting at 07:00pm as scripted. The evaluation team arrived in the place 4:00pm to all preparations. At 9:00pm, all evaluation activities had been concluded and the questionnaire answered.

Results of both evaluations are presented in next section.

4.5.3.3 Activity 4.3: Conclusion of the System Evaluation

Load Test Experiment

During the load test, the system behaved in a stable way, with good and constant response time observed when navigating through the ERTK screens and functions. However, ERTK stopped to show the received reports after about 2,000 reports had been consumed, which happened about two hours after the load test had started. Checking the ERTK console, it was possible to verify that the reports were still coming and being processed by the ERTK. After 10 hours of experiment, about 4,500 reports (out of 9,000) had been processed by ERTK. Therefore, the RESCUER SoS presented the capacity to successfully support the consumption of 2,000 reports, which took two hours. However, it is important to point out that the 2,000 reports were sent by JMeter about seven minutes after the start of the experiment. This means that, in two hour of execution, the RESCUER SoS was able to successfully process only reports sent in the first seven minutes of the simulated emergency situation. In others words, the capacity of the RESCUER SoS is 0.28 (2000/7200s) reports/seconds instead to be 2,5 (9000/3600s) as expected in average. Analysing these results, we consider that this capacity is not enough, as relevant information coming from the crowd would not be processed and visualised in the ERTK, which would impact the decision-making capacity of C&CC negatively.

Considering the information from the crowd must be available in the ERTK, the availability of RESCUER SoS was highly impacted due to ERTK problem reported abovementioned. However, the measurement of availability of RESCUER SoS can be realized in two ways: (i) considering that the expected availability of the SoS is only one hour (i.e., 60 minutes or the time taken to JMeter to send all reports), and (ii) considering that the expected availability is the total of the experiment time (10 hours or 600 minutes), i.e., until all reports sent to ERTK are consumed. Bellow, we present results considering both options:

Availability (1) = (Operation_time / Incident_time) * 100
= (60 min / 60 min) * 100
= 100%

Or

Availability (2) = (Operation_time/ Experiment_time) * 100
= (120 min[9] / 600 min) * 100
= 20%

RESCUER SoS must be available during the whole emergency situation and it must allow users to achieve their goals. Therefore, RESCUER SoS could be available during the whole emergency situation for the majority of the incidents at large-scale events and in industrial areas (two hour timeframe), but then the users would not be able to achieve their goals, because the data in the ERTK does not properly represent the current emergency state. For more complex and longer incidents, the availability of RESCUER SoS is not currently satisfactory. However, it is expected that RESCUER SoS availability has an improvement in its next version, since the problem reported (the main responsible for the current availability results) already was identified.

After 10 hours of experiment, about 4,500 reports (out of 9,000) had been processed by ERTK. Based on a simple forecast, ERTK would probably take about 20 hours to process 9,000 reports. According the RESCUER requirements specification (RESCUER,

[9] Time the ERTK stopped to show the received reports

2016c), RESCUER SoS should take up to three minutes to process a report with data analysis and one minute without analyzing data. This means that three minutes is the maximum expected delay for a report be consumed. Therefore, the response time of the current RESCUER SoS is still too far from ideal, as all reports should have already been processed by ERTK after 1h:03min (time the last report was sent + maximum expected delay) of experiment. In summary, the average time to process an incident report during the experiment was about eight seconds (36000s/45000 reports) instead of 0.42s (3780s/9000 reports).

The computer used to support the ERTK had 4GB memory and Xeon 2.4 GHz processor, Dual-Core, and Windows server 2012 R2 64 bits. During the experiment, the average percentage of the use of RAM and CPU were 15% and 1%, respectively. These results are very good regarding the system resource utilization efficiency, once only a small quantity of resources is requested by the RESCUER SoS currently.

In summary, analyzing results of load test experiment, we identified that some attention needs to be taken to ensure that quality attributes, such as availability, response time, and capacity, fulfill the RESCUER SoS requirements.

Emergency Simulation Exercise

About the emergency simulation exercise, the evaluation questionnaires of RESCUER SoS were answered by a total of four people distributed in four profiles: technical employee, rescue dog handler, rescue officer, and fireman. All of them were male and had experience in emergency situations. In average, participants had 40 years old. The complete results of the questionnaire can be seen in Table 4.12.

In summary, the most responders said the RESCUER SoS provides information in a timely manner to support making decision. All participants said that trust in RESCUER SoS to receive information from people in an incident place. However, it can be observed that most of them had doubt about the risk exposure of the people and the capacity of the SoS to reduce the use of resources in emergency situation, and if they would use or recommend the RESCUER SoS for emergency response or public authorities.

The small number of responders may not be enough to draw conclusive observations on the questions asked. However, they may hint on some feedback that could be necessary to advance on the specification of RESCUER SoS in the future. Therefore, the goal here is to make a brief assessment on the responses obtained during the evaluation. In this context, also it would not make sense to do a quantitative analysis of these results. However, we need to translate these feedbacks in quantitative values through the metrics application. This is important to the complete application of the SoS Evaluation Model described in Chapter 3, mainly regarding the application of the SoS aggregation defined in Section 3.6.

Table 4.12: RESCUER SoS questionnaire results

Question	**Technical employee**	**Rescue dog handler**	**Rescue officer**	**Fireman**
Q1. Does RESCUER provide its information in a timely manner to adequately support decision-making?	Yes	No	Yes	Yes
Q2. Can the use of RESCUER reduce the use of human and material resources needed to manage an emergency situation?	Maybe	Maybe	Certainly Yes	Maybe
Q3. Do you trust in RESCUER to receive information from people at the incident place?	Yes	Yes	Yes	Yes
Q4. Do you trust that information presented by RESCUER represents the real situation at the incident place?	Maybe	Yes	Yes	Maybe
Q5. Do you think the use of RESCUER in an emergency situation can expose people at the incident place to additional risks?	Maybe	No	Maybe	Yes
Q6. Would you actually use RESCUER for emergency management?	Maybe	Yes	No	Maybe
Q7. Would you recommend the use of RESCUER to other emergency response entities or public authorities?	Maybe	Yes	No	Maybe
Q8. Is there any other activity related to emergency management that RESCUER should support?	Yes	Yes	Yes	Yes

Therefore, a quantitative analysis was done from the questionnaires results, but they can not be considered as conclusive results of respective quality attributes. In Table 4.13, it is presented a summary of these quality attributes and their relation between questions used as source of data. In the case of more than one question contributing to a quality attribute, the final measure is the average of the results of each question.

After the conduction of both evaluation, results were compared to acceptance criteria established in the SoS Evaluation Plan. Table 4.14 presents the RESCUER SoS evaluation results of load test experiment (performance efficiency, and reliability) and emergency simulation exercise (functional suitability, usability, freedom from risk, and satisfaction). As it can be observed, all quality attributes evaluated during the emergency simulation exercise had no sufficient results. However, as explained before, they are not conclusive.

About the load test results, attention needs to be payed to capacity and time Behavior quality attributes, which were impacted by the ERTK problem previously reported. However, most of the problems found during this evaluation already were identified and some possible solutions proposed to be implemented in the next system version.

4.6 Activity 5: Conclusion of the SoS Evaluation

In this activity, firstly, results of each constituent system evaluation are put together to obtain the measurement of the whole quality of SoS. This activity can be conducted in the final of each project iteration as a way to better manage the quality of the SoS. In Sec-

Table 4.13: Relation between questions and metrics

Question	**Quality Attribute**	**Metric**
Q1. Does RESCUER provide its information in a timely manner to adequately support decision-making?	Efficiency	EF2
Q2. Can the use of RESCUER reduce the use of human and material resources needed to manage an emergency situation?	Efficiency	EF2
Q3. Do you trust in RESCUER to receive information from people at the incident place?	Trust, Usefulness	ST1, SU1
Q4. Do you trust that information presented by RESCUER represents the real situation at the incident place?	Trust, Usefulness	ST1, SU1
Q5. Do you think the use of RESCUER in an emergency situation can expose people at the incident place to additional risks?	Freedom from Risk	FR1
Q6. Would you actually use RESCUER for emergency management?	Usability (Appropriateness Recognisability), Usefulness, Trust	UA1, SU1, ST1
Q7. Would you recommend the use of RESCUER to other emergency response entities or public authorities?	Usability (Appropriateness Recognisability), Usefulness, Trust	UA1, SU1, ST1
Q8. Is there any other activity related to emergency management that RESCUER should support?	Functional Suitability (Completeness)	FF1

Table 4.14: RESCUER SoS evaluation results

Quality Attribute	**Quality Sub Attribute**	**Metric**	**Accept. Criteria**	**Measurement**	**Results**
Performance Efficiency	Resource Utilization	PR3	0.8	0.85	Yes
		PR2	0.8	0.99	Yes
	Capacity	PC1	2.5	0.28	No
	Time Behavior	PT1	0,42	8.0	No
Reliability	Availability	RA1	0.9	1.0	Yes
Functional Suitability	Completeness	FF1	0.8	0.0	No
Usability	Appropriateness Recognisability	UA1	0.8	0.25	No
Freedom from Risk		FR1	0.8	0.5	No
Satisfaction	Usefulness	SU1	0.8	0.5	No
	Trust	ST1	0.8	0.25	No

tion 4.5, we reported the planning, conduction, and results of MCS evaluation conducted in first project iteration, ERTK evaluation conducted in second project iteration, and RESCUER SoS evaluation conducted in third project iteration. In this section, results of all evaluation conducted in the context of RESCUER project are considered. Table 4.15 presents the list of all evaluations conducted during the RESCUER project with the country, place, number of participants, and evaluation strategy considered. These evaluations took place in four different countries, Brazil, Germany, Spain, and Austria, using three

different evaluation strategies: emergency simulation exercise, evaluation survey, and load test experiment.

Table 4.15: Evaluations conducted during RESCUER project

Project Iteration	**Constit. System**	**Country**	**Place**	**Part.**	**Evaluation Strategy**
First Iteration	MCS	Brazil	FIFA FAN Fest (Salvador)	35	Evaluation Survey
			University of São Paulo (São Carlos)	27	Evaluation Survey
			COFIC - Industrial Park (Camaçari)	50	Simulation Exercise
		Germany	Fritz Walter Stadium (Kaiserslautern)	60	Evaluation Survey
	ERTK	Brazil	CICC - Integrated Command and Control Centre (Salvador); Bahia's Civil Police; COFIC - Industrial Park (Camaçari)	4	Evaluation Survey
		Spain	CISEM - Centro Integrado de Seguridad y Emergencias (Madrid)	1	Evaluation Survey
		Austria	Fire Brigade Chemical Park (Linz); Austrian Fire Brigade (Linz); Austrian Red Cross (Linz)	6	Evaluation Survey
Second Iteration	MCS	Brazil	CIDEMII - International Congress on Mass Disasters (Salvador)	31	Evaluation Survey
			COFIC - Industrial Park (Camaçari)	24	Simulation Exercise
		Germany	Fraunhofer IESE (Kaiserslautern)	22	Experience
	ERTK	Brazil	CIDEMII - International Congress on Mass Disasters (Salvador); CEIC - Integrated Command Center (Porto Alegre)	15	Evaluation Survey
Third Iteration	MCS	Austria	Lambach Tunnel (Linz), Stadion der Stadt (Linz)	32	Simulation Exercise
	ERTK		Lambach Tunnel (Linz)	4	Simulation Exercise
	SoS	Austria	Lambach Tunnel (Linz)	4	Simulation Exercise
		Brazil	Load Test (São Carlos)	-	Experience

In summary, emergency simulation exercises were used to simulate an incident in a controlled way to provide a more realistic incident scenario as possible. Despite the need of a bigger effort to plann and conduct this kind of strategy, it was very important to exercise the whole system in an integrated way, besides to check the influence of the participant's stress that is expected in a real emergency situation. In the load test experiment no final users were involved. The main goal was to simulate thousands of users of MCS application sending incident reports, while the RESCUER SoS was monitored. Evaluation survey was used as strategy to obtain opinion of final users and technical experts about key aspects of the system. Using this strategy, participants were asked to use the system considering an incident scenario, and their opinion catched through a questionnaire application, basically.

Despite constituent systems DAS and COM were integrated in the whole RESCUER SoS, they were not formally evaluated during RESCUER project. Because this, Table 4.15 does not mention these constituent systems. However, this did not represent any loss for the application of the SoS Evaluation Model, since that evaluation of two constituent system and RESCUER SoS was enough to the complete application of this model and, mainly, to the aggregation of results to obtain values representing the measurement of the RESCUER SoS quality.

Table 4.16, Table 4.17, and Table 4.18 present the application of SoS aggregation model in the three project iterations, respectively. In this model, four aggregation levels are needed to obtain the measure of whole quality of RESCUER SoS. In first aggregation level (represented by column "A1"), results from each metric were aggregated to obtain the quantitative value, which refers to degree of compliance of constituent system to the corresponding quality sub attribute. In second aggregation level (represented by column "A2"), results of each quality sub attribute were aggregated to obtain the value for the quality attributes. In third aggregation level (represented by column "A3"), the value that refers the quality of each constituent system was obtained through the aggregation of results of all quality attributes. Finally, in fourth aggregation (represented by column "A4"), whole quality of RESCUER SoS is obtained by aggregation of results of each constituent system. This process is applied to each project iteration.

Table 4.16: Results aggregation in first project iteration

A4 - SoS Quality	**Const. System**	**Prior.**	**A3**	**Qualiy Attribute**	**Prior.**	**A2**	**Quality Sub Attribute**	**Prior.**	**A1**	**Metric**	**Result**
0.77	MCS	3	0.81	Usability	3	0.79	User Interface Aesthetics	3	0.86	UU1	0.86
							Learnability	3	0.71	UL1	0.64
										UL2	0.77
				Effectiveness	3	0.71	Effectiveness	3	0.71	EF1	0.71
				Satisfaction	3	0.93	Usefulness	3	0.96	SU1	0.96
							Trust	3	0.90	ST1	0.90
	ERTK	3	0.73	Usability	2	0.60	User Interface Aesthetics	2	0.82	UU1	0.82
							Appropriateness Recognisability	2	0.17	UA1	0.17
							Learnability	1	1.0	UL1	1.0
				Satisfaction	3	0.82	Usefulness	3	0.82	SU1	0.82

It is important to highlight that, for the correct aggregation of metrics results, it is needed to respect a same data scale. Most metrics used in the RESCUER quality measurement produce a quantitative value between 0 and 1. Therefore, metric results that do not respect this scale needed to be normalized. This was done to results of metrics PC1 and PT1, regarding capacity and time behavior, respectively (compare Table 4.14 and Table 4.18).

All aggregation levels were calculate using Equation 3.1 presented in Section 3.6. In first aggregation, no weights were considered, as any priority to the metrics was not assigned . In second aggregation, weight considered to each quality sub attribute was obtained from the RESCUER quality model presented in Section 4.2. During the construction of this quality model, the priority of each quality attribute to the achievement of the system goals were collected from the RESCUER stakeholders as H - High (weight 3), M - Medium (weight 2), and L - Low (weight 1). In third aggregation, weight of

Table 4.17: Results aggregation in second project iteration

A4 - SoS Quality	**Const. System**	**Prior.**	**A3**	**Quality Attribute**	**Prior.**	**A2**	**Quality Sub Attribute**	**Prior.**	**A1**	**Metric**	**Result**
0.91	MCS	3	0.86	Functional Suitability	3	0.71	Completeness	2	0.66	FF1	0.66
							Appropriateness	3	0.75	FF2	0.75
				Usability	3	0.83	User Interface Aesthetics	3	0.80	UU1	0.80
							Appropriateness Recognisability	3	0.91	UA1	0.91
							Learnability	3	0.81	UL3	0.81
							Operability	3	0.78	UO2	0.78
				Portability	3	1.0	Portability	3	1.0	PI1	1.0
				Freedom from Risk	3	0.71	Freedom from Risk	3	0.71	FR1	0.71
				Satisfaction	3	0.98	Usefulness	3	0.98	SU1	0.98
							Trust	3	0.98	ST1	0.98
				Effectiveness	3	0.91	Effectiveness	3	0.91	EF1	0.91
	ERTK	3	0.95	Usability	2	0.94	Operability	2	0.93	UO2	0.93
							Appropriateness Recognisability	2	0.94	UA1	0.94
							Learnability	1	0.94	UL3	0.94
				Effectiveness	3	0.98	Effectiveness	3	0.98	EF1	0.98
				Efficiency	2	0.87	Efficiency	2	0.87	EF2	0.87
				Satisfaction	3	0.97	Usefulness	3	0.97	SU1	0.97

Table 4.18: Results aggregation in third project iteration

A4 - SoS Quality	**Const. System**	**Prior.**	**A3**	**Quality Attribute**	**Prior.**	**A2**	**Quality Sub Attribute**	**Prior.**	**A1**	**Metric**	**Result**
0.57	MCS	3	0.70	Functional Suitability	3	0.90	Completeness	2	0.94	FF1	0.94
							Appropriateness	3	0.87	FF2	0.87
				Usability	3	0.71	Appropriateness Recognisability	3	0.70	UA1	0.70
							User Interface Aesthetics	3	0.71	UU1	0.71
							Learnability	3	0.77	UL3	0.77
							Operability	3	0.67	UO2	0.67
				Freedom from Risk	3	0.5	Freedom from Risk	3	0.5	EF1	0.5
				Satisfaction	3	0.70	Usefulness	3	0.70	SU1	0.70
							Trust	3	0.70	ST1	0.70
	ERTK	3	0.45	Usability	2	0.38	Appropriateness Recognisability	2	0.38	UA1	0.38
							Operability	2	0.38	UO2	0.38
				Effectiveness	3	0.38	Effectiveness	3	0.38	EF1	0.38
				Efficiency	2	0.25	Efficiency	2	0.25	EF2	0.25
				Satisfaction	3	0.69	Usefulness	3	0.75	SU1	0.75
							Trust	3	0.63	ST1	0.63
0.44	SoS	3	0.44	Functional Suitability	2	0.0	Completeness	2	0.0	FF1	0.0
				Usability	3	0.25	Appropriateness Recognisability	3	0.25	UA1	0.25
				Performance Efficiency	2	0.36	Resource Utilization	2	0.92	PR3	0.85
										PR2	0.99
							Capacity	2	0.11	PC1	0.11
							Time Behavior	2	0.05	PT1	0.05
				Reliability	3	1.0	Availability	3	1.0	RA1	1.0
				Freedom from Risk	3	0.5	Freedom from Risk	3	0.5	FR1	0.5
				Satisfaction	3	0.38	Usefulness	3	0.5	SU1	0.5
							Trust	3	0.25	ST1	0.25

each quality attribute was obtained by the average of the weights of its quality sub attributes. In the final aggregation, for both constituent systems, the maximum priority was assigned, since they are the most important constituent systems of the RESCUER SoS. These weights could also be obtained by a survey with stakeholders.

After the application of the aggregation model, we obtained values that represent the whole quality of RESCUER SoS during the project. These values were 0.77 in first iteration, 0.91 in second iteration, and 0.57 in third iteration. This quality management is very important to verify the evolution of SoS quality. A good improvement of the SoS quality can be noticed in the second iteration, but the same improvement did not happen in third iteration. This information allowed us to identify the reasons for that, and possible points of improvement in the RESCUER SoS.

Actually, the SoS quality in third iteration was very impacted by poor results of evaluation conducted in Lambach tunnel, where only four people participated and, therefore, the results were not considered conclusive. For this reason, the SoS quality measured in third iteration did not represent the real RESCUER quality. However, even so we used these results to allow the complete application of the SoS Evaluation Model defined in this Master's project.

As explained in Section 3.6, it is possible that two evaluation results of SoS be produced: one obtained from the aggregation of results of all constituent systems and another from the evaluation of the SoS as an independent system. These values can be seen in the results of third iteration (see Table 4.18), where specific quality metrics were applied to the SoS. This complementary result allows to identify problems in the quality attributes that only become evident when all constituents are working together. The resulting value of the evaluation of RESCUER SoS in third iteration was 0.44. This value was also negatively impacted by the poor results of load test experiment, mainly about the capacity and time behavior.

In summary, the proposed SoS aggregation model was successfully applied in RESCUER context, in all its aggregation levels. This model can manage the SoS quality, considering the importance of the quality value produced by composition of constituent system quality, and the SoS quality obtained through its evaluation as an independent system.

4.7 Final Remarks

This chapter presented a case study of application of the SoS Evaluation Model described in Chapter 3 in the context of RESCUER research project. In this sense, our experience regarding the planning, execution, and conclusion of the RESCUER project evaluations was reported. To describe the application of this model, three evaluations (out of 20) were reported in detail based on the SoS Evaluation Plan, which was produced during the planning activities of SoS evaluation.

Due to the level of operational and managerial independence of the constituent organizations, ensure that all evaluations were conducted considering the SoS Evaluation Plan was a bit difficult. In summary, the evaluations tended to be planned, conducted, and mainly reported in a no standardized way. This can becomes difficult to put together all evaluation results to perform the SoS quality aggregation. In addition, some times, constituent system evaluations were planned and conducted without to consider the application of all metrics and quality attributes defined into SoS Evaluation Plan. This happened in some cases, by carelessness, or when the evaluation context becomes difficult to apply evaluation strategies that would allow to obtains the needed data to application of specifics metrics. In both situations, the SoS Evaluation Plan was updated after the evaluations be performed to maintain the coherence. In short, this model was successfully applied in RESCUER context, showing that is possible to effectively manage the quality of a SoS through its project iterations.

Chapter 5

Conclusions

SoS is becoming increasingly important and being applied in several critical sectors of the society. This class of systems emerges as a result of the interaction of large, complex, and independent systems. By their criticality, evaluation of their quality is essential. However, the quality management of SoS still presents great challenges, as these systems have a set of unique characteristics when compared to monolithic systems. In addition, there are no specific studies or wide experience reports to guide the quality evaluation of SoS, considering all of its characteristics and mainly the challenges inherent of this context.

5.1 Contribution of this Master's Project

This Master's project presented a SoS Evaluation Model, addressed to the crisis/emergency management domain. The main goal of this model is to cover important evaluation activities, considering the main SoS characteristics and challenges imposed by these systems still not addressed by current models. The SoS Evaluation Model was built considering recommendations and guidelines provided by the main international standards focused on software quality and also our experience in planning, executing, and reporting SoS quality evaluations. In this sense, most important software quality evaluation activities were reviewed and restructured to both constituent system and SoS levels. Besides that, guidelines and recommendations were provided for all evaluation activities to allow the suitable adoption of this model.

This SoS Evaluation Model was applied in the context of a large international research project entitled RESCUER, which aimed to develop a crisis/emergency management SoS. Our model was used as basis to evaluation of the RESCUER SoS and its main constituent systems during three project iterations. A total of 20 evaluations were conducted during the project, allowing us to verify the suitability of our model, besides obtaining a wide experience that was used to refine the model. In short, the proposed SoS Evaluation Model was successfully applied in RESCUER context in all its project iteration.

By applying our model, a set of artifacts are created, including a domain specific quality model, a set of metrics, evaluation and planning strategies to both SoS level and constituents level. In particular, all artifacts produced applying our model in the RESCUER SoS evaluation could be used as basis to conduction of evaluations of any other crisis/emergency management SoS. In a more generic way, this model could be reused in any acknowledged SoS, which have recognized goals, a management team, and own resources, despite the constituent systems retain their managerial and operational independence.

Our experience applying the SoS Evaluation Model showed us that it is possible to effectively manage the quality of a SoS throughout of its project iterations. This was possible through the measurement of the SoS quality by considering the quality of all its constituent systems, and also the evaluation of the SoS as an independent system. In general, these measurements allow the management of quality attributes that only become evident when all constituents are working together and, therefore, they would not be measured only by evaluating individually the constituent systems. On the other hand, the composition of the evaluation results supported us to identify problems with constituent systems that have negatively impacted the whole quality of this SoS.

It is important to highlight that SoS have been increasingly adopted in different and critical application domains. We believe that our model can be a good starting point and basis to further initiatives to develop or improve the SoS Evaluation Model for other applications domains.

5.2 Difficulties and Limitations

As this model was developed in the context of a specific research project, evidences still need to be raised about its suitability for other types of SoS, application domains, even so to others crisis/emergency management SoS. Therefore, due to the wide possibility of organizational structures, roles and dependencies among the parts that compose a SoS, it is possible that adaptations need to be done in the proposed model. Despite to be used to conduct a total of 20 evaluations in the context of RESCUER project, this evaluation model was applied in only one case study. This was due to the time constraints to the conduction of the Master's project.

Difficulties and limitations were also found during the conduction of our case study. It was clear that the level of operational and managerial independence of the constituent organizations, geographic distribution, and the evolutionary development adopted in RESCUER project were responsible for the main difficulties and limitations found during the application of our evaluation model. In this scenario, maintaining the SoS Evaluation Plan updated regarding quality requirements in constant changing, and also coherent with regard to what were actually conducted by constituent organizations was a difficult task.

5.3 Future Works

This Master's project has opened several perspectives of research. As main future works that we intend to develop are:

- To conduct other case studies and experiments to verify the suitability of our model in other crisis and emergencies management SoS. The main idea is, firstly, to verify the suitability and coverage of our model in similar SoS contexts, considering variations in their organizational structures, roles, and dependencies between constituent systems. In addition, we intend to verity the coverage and suitability of artifacts developed during the conduction of this Master's project, when used to evaluate other crisis and emergencies management SoS. New strategies and techniques could be also added to this model to facilitate the tailoring of these artifacts;
- To apply our model in other SoS application domains (e.g., robotics, ambient assisted living - AAL, IT systems, etc.) with the goal of, besides adapting and refining the model, developing a new set of artifact, such as domain specific quality model, set of metrics, evaluation and planning strategies, which could be used as basis to evaluate other SoS domains;
- To conduct surveys with software quality and SoS experts from the industry, mainly to validate and refine the overall structure adopted in this model, as well as the recommendations and guidelines to become them even more aligned to best practices and real needs of a wider number of evaluation initiatives addressed to complex, distributed SoS development projects; and
- Extend our model to other types of SoS, i.e., directed, collaborative, and virtual. These types of SoS have specific characteristics based upon their organizational

structure and level of responsibility and authority of the constituent systems regarding the SoS missions. Therefore, these aspects and characteristics need to be analyzed and understood to develop a broader SoS evaluation model that could be suitable to evaluate all types of SoS.

References

Ackermann, C.; Lindvall, M.; Cleaveland, R. Towards behavioral reflexion models. In: *20th International Symposium on Software Reliability Engineering (ISSRE)*, 2009, p. 175–184.

AL-Badareen, A.; Selamat, M.; A. Jabar, M.; Din, J.; Turaev, S. Software quality models: A comparative study. In: *2th International Conference Software Engineering and Computer Systems (ICSECS)*, 2011, p. 46–55.

Alghamdi, A.; Hussain, T.; Faraz Khan, G. Enhancing c4i security using threat modeling. In: *12th International Conference on Computer Modelling and Simulation (UKSim)*, 2010, p. 131–136.

Allen, M. From substandard to successful software. *CrossTalk: The Journal of Defense Software Engineering*, v. 22, n. 4-5, p. 29–32, 2009.

Aoyama, M.; Tanabe, H. A design methodology for real-time distributed software architecture based on the behavioral properties and its application to advanced automotive software. In: *18th Asia-Pacific Software Engineering Conference (APSEC)*, 2011, p. 211–218.

Azizian, N.; Mazzuchi, T.; Sarkani, S.; Rico, D. A framework for evaluating technology readiness, system quality, and program performance of u.s. dod acquisitions. *Systems Engineering*, v. 14, n. 4, p. 410–426, 2011.

Babar, M.; Zhu, L.; Jeffery, R. A framework for classifying and comparing software architecture evaluation methods. In: *15th Australian Software Engineering Conference (ASWEC)*, 2004, p. 309–318.

Babar, M. A.; Chauhan, M. A. A tale of migration to cloud computing for sharing experiences and observations. In: *2th International Workshop on Software Engineering for Cloud Computing (SECLOUD)*, 2011, p. 50–56.

Balci, O.; Arthur, J. D.; Ormsby, W. F. Achieving reusability and composability with a simulation conceptual model. *Journal of Simulation*, v. 5, n. 3, p. 157–165, 2011.

Barcelos, R.; Travassos, G. Arqcheck: Uma abordagem para inspeção de documentos arquiteturais baseada em checklist. *Simposio Brasileiro de Qualidade de Software (SBQS)*, p. 175–189, 2006.

Barron, F. H.; Barrett, B. E. Decision quality using ranked attribute weights. *Management Science*, v. 42, n. 11, p. 1515–1523, 1996.

Batista, T. Challenges for SoS Architecture Description. In: *2th International Workshop on Software Engineering for Systems-of-Systems (SESoS)*, 2013, p. 35–37.

Belloir, N.; Chiprianov, V.; Ahmad, M.; Munier, M.; Gallon, L.; Bruel, J.-M. Using Relax Operators into an MDE Security Requirement Elicitation Process for Systems of Systems. In: *European Conference on Software Architecture Workshops (ECSAW)*, 2007, p. 32:1–32:4.

Bianchi, T.; Santos, D. S.; Felizardo, K. R. System-of-Systems Quality Attributes: A Systematic Literature Review. In: *3th International Workshop on Software Engineering and Systems-of-Systems (SESoS)*, 2015, p. 23–30.

Boehm, B.; Lane, J. A. 21st Century Processes for Acquiring 21st Century Software - Intensive Systems of Systems. *Crosstalk: Journal of Defence Software Engineering*, v. 19, n. 5, p. 4–9, 2006.

Boehm, B. W.; Brown, J. R.; Kaspar, J. R.; Lipow, M. L.; MacCleod, G. Characteristics of software quality. *Elsevier*, 1978.

Bozzano, M.; Villafiorita, A. *Design and safety assessment of critical systems.* Taylor & Francis, 2010.

Calinescu, R.; Kikuchi, S.; Johnson, K. Compositional Reverification of Probabilistic Safety Properties for Large-scale Complex IT Systems. In: *17th Monterey Conference on Large-Scale Complex IT Systems: Development, Operation and Management*, 2012, p. 303–329.

Chen, H.-M.; Kazman, R. b.; Hong-Mei, C. Architecting ultra-large-scale green information systems. In: *1th International Workshop on Green and Sustainable Software (GREENS)*, 2012, p. 69–75.

Chiprianov, V.; Falkner, K.; Gallon, L.; Munier, M. Towards modelling and analysing non-functional properties of systems of systems. In: *9th International System of Systems Engineering Conference (SOSE)*, 2014, p. 289–294.

Clements, P.; Kazman, R.; Klein, M. *Evaluating software architectures: methods and case studies*. SEI series in software engineering. Addison-Wesley, 2002.

Dahmann, J.; Baldwin, K. Understanding the current state of us defense systems of systems and the implications for systems engineering. In: *2th IEEE Systems Conference*, 2008, p. 1–7.

Dieste, O.; Grimán, A.; Juristo, N. Developing search strategies for detecting relevant experiments. *Empirical Software Engineering*, v. 14, n. 5, p. 513–539, 2009.

DoD *Systems engineering guide for system of systems*. Technical Report, US Department of Defense, 2008.

DoD *DoDAF Architecture Framework Version 2.02*. Technical Report, US Department of Defense, 2010.

Dybå, T.; Dingsøyr, T. Strength of evidence in systematic reviews in software engineering. In: *2th International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2008, p. 178–187.

Edwards, W.; Barron, F. SMARTS and SMARTER: Improved Simple Methods for Multiattribute Utility Measurement. *Organizational Behavior and Human Decision Processes*, v. 60, n. 3, p. 306 – 325, 1994.

Eisner, H. Rcasse: Rapid computer-aided systems of systems engineering. In: *International Symposium of the National Council of System Engineering (NCOSE)*, 1993, p. 267–273.

Eklund, U. c.; Bosch, J. Architecture for embedded open software ecosystems. *Journal of Systems and Software*, v. 92, n. 1, p. 128–142, 2014.

Fang, Z.; DeLaurentis, D.; Davendralingam, N. An approach to facilitate decision making on architecture evolution strategies. *Procedia Computer Science*, v. 16, n. 0, p. 275 – 282, 2013.

Fischer, D.; Sarkarati, M.; Spada, M.; Michelbach, T.; Urban, W.; Tueffers, C. An application security framework for soa-based mission data systems. In: *4th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 2011, p. 53–60.

Fuchs, A.; Rieke, R. Identification of security requirements in systems of systems by functional security analysis. In: Casimiro, A.; de Lemos, R.; Gacek, C., eds. *Architecting Dependable Systems VII*, v. 6420, p. 74–96, 2010.

Gagliardi, M.; Wood, W.; Klein, J.; Morley, J. A uniform approach for system of systems architecture evaluation. *The Journal of Defense Software Engineering*, v. 22, n. 3, p. 12–15, 2009.

Garro, A.; Tundis, A. On the Reliability Analysis of Systems and SoS: The RAMSAS Method and Related Extensions. *IEEE Systems Journal*, v. PP, n. 99, p. 1–10, 2014.

Garvin, D. What does product quality really mean? *Sloan Management Review*, v. 26, n. 1, p. 25–45, 1984.

Ge, B.; Hipel, K. W.; Yang, K.; Chen, Y. A data-centric capability-focused approach for system-of-systems architecture modeling and analysis. *Systems Engineering*, v. 16, n. 3, p. 363–377, 2013.

Gorod, A.; Gove, R.; Sauser, B.; Boardman, J. System of Systems Management: A Network Management Approach. In: *2th International Conference on System of Systems Engineering (SoSE)*, 2007, p. 1–5.

Grady, R. B.; Caswell, D. L. *Software Metrics: Establishing a Company-wide Program.* Prentice-Hall, Inc., 1987.

Guariniello, C.; DeLaurentis, D. Communications, Information, and Cyber Security in Systems-of-Systems: Assessing the Impact of Attacks through Interdependency Analysis. *Procedia Computer Science*, v. 28, p. 720 – 727, 2014a.

Guariniello, C.; DeLaurentis, D. Integrated analysis of functional and developmental interdependencies to quantify and trade-off ilities for system-of-systems design, architecture, and evolution. *Procedia Computer Science*, v. 28, p. 728–735, 2014b.

Guo, F.; Wang, M. Quantitative measurement of interoperability by using Petri net. *Journal of Computational Information Systems*, v. 8, n. 8, p. 3245–3252, 2012.

Iacobucci, J.; Mavris, D. A method for the generation and evaluation of architecture alternatives on the cloud. In: *6th System of Systems Engineering (SoSE)*, 2011, p. 137–142.

ISO/IEC 24765 *Systems and software engineering - Vocabulary.* Technical Report, International Organization for Standardization and International Electrotechnical Commission, 2010.

ISO/IEC 25010 *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.* Technical Report, International Organization for Standardization and International Electrotechnical Commission, 2011.

ISO/IEC 25040 *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Evaluation process.* Technical Report, International Organization for Standardization and International Electrotechnical Commission, 2011.

ISO/IEC 9126-2 *Software engineering - Product quality - Part 2: External metrics.* Technical Report, International Organization for Standardization and International Electrotechnical Commission, 2003.

ISO/IEC 9126-3 *Software engineering - Product quality - Part 3: Internal metrics.* Technical Report, International Organization for Standardization International Electrotechnical Commission, 2003.

ISO/IEC 9126-4 *Software engineering - Product quality - Part 4: Quality in Use metrics.* Technical Report, International Organization for Standardization International Electrotechnical Commission, 2003.

Jamshidi, M. *Systems of Systems Engineering: Principles and Applications.* Taylor & Francis, 2008.

Joyce, H.; Hin, O.; Seng, Y.; Chan, S. A systems assurance perspective towards generic systems engineering. In: *21th International Symposium of the International Council on Systems Engineering (INCOSE)*, 2011, p. 742–761.

Kazman, R.; Bass, L.; Webb, M.; Abowd, G. Saam: A method for analyzing the properties of software architectures. In: *16th International Conference on Software Engineering (ICSE)*, 1994, p. 81–90.

Kazman, R.; Gagliardi, M.; Wood, W. Scaling up software architecture analysis. *Journal of Systems and Software*, v. 85, n. 7, p. 1511 – 1519, 2012.

Kazman, R.; Klein, M.; Clements, P. *ATAM: Method for Architecture Evaluation.* Technical Report, Carnegie Mellon University, 2000.

Kimura, D.; Osaki, T.; Yanoo, K.; Izukura, S.; Sakaki, H.; Kobayashi, A. Evaluation of IT systems considering characteristics as system of systems. In: *6th International Conference on System of Systems Engineering (SoSE)*, 2011, p. 43–48.

Kitchenham, B.; Charters, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering.* Technical Report, Keele University and Durham University Joint Report, 2007.

Kitchenham, B.; Pfleeger, S. L. Software quality: The elusive target. *IEEE Software*, v. 13, n. 1, p. 12–21, 1996.

Lane, J. A. *What is a system of systems and why should i care?* Technical Report, Department of Industrial and Systems Engineering - University of Southern California, 2013.

Larsson, J.; Borg, M.; Olsson, T. Testing quality requirements of a system-of-systems in the public sector - challenges and potential remedies. In: *3rd International Workshop on Requirements Engineering and Testing (RET)*, 2016, p. 1–15.

Leuchter, S.; Reinert, F.; Muller, W. Assessment of the integration capability of system architectures from a complex and distributed software systems perspective. In: *SPIE Open Architecture/Open Business Model Net-Centric Systems and Defense Transformation*, 2014, p. 4–9.

Lin, Q.; Cai, Z.; Wang, Y.; Yang, J.; Chen, L. Adaptive Flight Control Design for Quadrotor UAV Based on Dynamic Inversion and Neural Networks. In: *3th Iternational Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC)*, 2013, p. 1461–1466.

Luzeaux, D.; Ruault, J. *Systems of Systems.* ISTE. Wiley, 2010.

Madni, A.; Sievers, M. System of systems integration: Key considerations and challenges. *Systems Engineering*, v. 17, n. 3, p. 330–347, 2014.

Maier, M. W. Architecting principles for systems-of-systems. *Systems Engineering*, v. 1, p. 267–284, 1998.

Mårtensson, F.; Tekniska, B. h. *Software Architecture Quality Evaluation: Approaches in an Industrial Context.* Blekinge Institute of Technology licentiate dissertation series. 2006.

McCall, J. A.; Richards, P. K.; Walters, G. F. Factors in software quality. *Rome Air Development Center Air Force Systems Command*, 1977.

Meilich, A. System of systems (SoS) engineering amp; architecture challenges in a net centric environment. In: *2006 IEEE/SMC International Conference on System of Systems Engineering*, 2006, p. 1–5.

Michael, J.; Riehle, R.; Shing, M.-T. The verification and validation of software architecture for systems of systems. In: *4th IEEE International Conference on System of Systems Engineering (SoSE)*, 2009, p. 1–6.

Minkiewicz, A. F. Tackling the cost challenges of system of systems. *The Journal of Defense Software Engineering*, v. 19, n. 5, p. 10–14, 2006.

Nair, S.; De La Vara, J.; Sabetzadeh, M.; Briand, L. An extended systematic literature review on provision of evidence for safety certification. *Information and Software Technology*, v. 56, n. 7, p. 689–717, 2014.

Nakagawa, E. Y.; Gonçalves, M.; Guessi, M.; Oliveira, L. B. R.; Oquendo, F. The state of the art and future perspectives in systems of systems software architectures. In: *1th International Workshop on Software Engineering for Systems-of-Systems (SESoS)*, 2013, p. 13–20.

Oberndorf, P.; Sledge, C. Evolution of a software engineer in a SoS system engineering world. In: *4th Annual IEEE Systems Conference*, 2010, p. 91–96.

Oliveira, M.; Pereira, J. Extensible Virtual Environment Systems Using System of Systems Engineering Approach. In: *17th Artificial Reality and Telexistence (ICAT)*, 2007, p. 89–96.

Osmundson, J. S.; Langford, G. O. Connections in System of Systems. *INCOSE International Symposium*, v. 22, n. 1, p. 787–799, 2012.

Ramaswamy, A.; Monsuez, B.; Tapus, A. Formal models for cognitive systems. In: *16th International Conference on Advanced Robotics (ICAR)*, 2013, p. 1–8.

Ravichandran, T.; Rothenberger, M. A. Software reuse strategies and component markets. *Communications of the ACM*, v. 46, n. 8, p. 109–114, 2003.

RESCUER *D5.2.1 - Evaluation Report of Mobile Crowdsourcing Solution 1.* Technical Report, Project RESCUER, 2015.

RESCUER *D4.4.3 - Emergency Response ToolKit 3.* Technical Report, Project RESCUER, 2016a.

RESCUER *D5.4.2 - Evaluation Report of Integrated Solution 2.* Technical Report, Project RESCUER, 2016b.

RESCUER *D7.4 - System Specification.* Technical Report, Project RESCUER, 2016c.

Rezaei, R.; Chiew, T.; Lee, S. An interoperability model for ultra large scale systems. *Advances in Engineering Software*, v. 67, p. 22–46, 2014.

Rothenberg, J. Interoperability as a cross-cutting concern. In: *Interoperabiliteit: Eerlijk zullen we alles delen*, RAND Corporation, 2008, p. 1–12.

Sage, A. P.; Cuppan, C. D. On the Systems Engineering and Management of Systems of Systems and Federations of Systems. *Information Knowledge Systems Management*, v. 2, n. 4, p. 325–345, 2001.

Santos, D. S.; Oliveira, B.; Duran, A.; Nakagawa, E. Reporting an experience on the establishment of a quality model for systems-of-systems. In: *27th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 2015a, p. 304–309.

Santos, D. S.; Oliveira, B.; Guessi, M.; Oquendo, F.; Delamaro, M.; Nakagawa, E. Y. Towards the evaluation od system of systems softaware architecture. *8th Workshop em Desenvolvimento Distribuido de Software, Ecossistemas de Software e Sistema de Sistemas (WDES)*, p. 1–6, 2014.

Santos, D. S.; Oliveira, B.; Nakagawa, E. Evaluation of a Crowdsourcing System: An experience report. In: *1th Workshop on Crowdsourcing Systems (SCrowd)*, 2015b, p. 29–36.

Schneider, D.; Trapp, M. Runtime Safety Models in Open Systems of Systems. In: *28th Digital Avionics Systems (DASC)*, 2009, p. 455–460.

Schneider, D.; Trapp, M. A Safety Engineering Framework for Open Adaptive Systems. In: *5th Self-Adaptive and Self-Organizing Systems (SASO)*, 2011, p. 89–98.

Schugerl, P.; Rilling, J.; Witte, R.; Charland, P. A Quality Perspective of Software Evolvability Using Semantic Analysis. In: *3th International Conference on Semantic Computing (ICSC)*, 2009, p. 420–427.

Shukla, M.; Asundi, J. Considering emergency and disaster management systems from a software architecture perspective. *International Journal of System of Systems Engineering (SoSE)*, v. 3, n. 2, p. 129–141, 2012.

Singh, A.; Dagli, C. H. Multi-objective stochastic heuristic methodology for tradespace exploration of a network centric system of systems. In: *3th Annual IEEE Systems Conference*, 2009, p. 218–223.

Stratton, W.; Sibol, D.; Lindvall, M.; Ackermann, C.; Godfrey, S. Developing an approach for analyzing and verifying system communication. In: *IEEE Aerospace Conference (AeroConf)*, 2009a, p. 1–13.

Stratton, W. C.; Sibol, D. E.; Lindvall, M.; Ackermann, C.; Godfrey, S. Developing an approach for analyzing and verifying system communication. In: *IEEE Aerospace Conference (AeroConf)*, 2009b, p. 1–13.

Tsadimas, A.; Kapos, G.-D.; Dalakas, V.; Nikolaidou, M.; Anagnostopoulos, D. Integrating simulation capabilities into SysML for enterprise information system design. In: *9th System of Systems Engineering (SOSE)*, 2014, p. 272–277.

Tucker, A.; Dagli, C. Design of experiments as a means of lean value delivery to the flight test enterprise. *Systems Engineering*, v. 12, n. 3, p. 201–217, 2009.

Urwin, E.; Venters, C.; Russell, D.; Liu, L.; Luo, Z.; Webster, D.; Henshaw, M.; Xu, J. Scenario-based design and evaluation for capability. In: *5th International Conference on System of Systems Engineering (SoSE)*, 2010, p. 1–6.

Väätäjä, H.; Koponen, T.; Roto, V. Developing practical tools for user experience evaluation: a case from mobile news journalism. In: *1th Energy Conversion Congress and Exposition (ECCE)*, 2009, p. 23.

Venters, C.; Russell, D.; Liu, L.; Luo, Z.; Webster, D.; Xu, J. A scenario-based architecture evaluation framework for network enabled capability. In: *33th Computer Software and Applications Conference (COMPSAC)*, 2009, p. 9–12.

Villalba, M. T.; Fernandez-Sanz, L.; Martinez, J. J. Empirical support for the generation of domain-oriented quality models. *IET Software*, v. 4, n. 1, p. 1–14, 2010.

Villela, K.; Vieira, V.; Mendonca, M.; Franke, T.; Torres, J.; Graffy, S. RESCUER-Dow - Reliable and Smart Crowdsourcing Solution for Emergency and Crisis Management. 2013.

Wada, H.; Suzuki, J.; Oba, K. A model-driven development framework for non-functional aspects in service oriented architecture. *International Journal of Web Services Research (IJWSR)*, v. 5, n. 4, p. 1–31, 2008.

Wagner, S. *Software Product Quality Control.* Berlin, Heidelberg: Springer, 2013.

Wagner, S.; Lochmann, K.; Heinemann, L.; Kläs, M.; Trendowicz, A.; Plösch, R.; Seidl, A.; Goeb, A.; Streit, J. The Quamoco Product Quality Modelling and Assessment Approach. In: *34th International Conference on Software Engineering (ICSE)*, 2012, p. 1133–1142.

Waller, A.; Craddock, R. Managing runtime re-engineering of a System-of-Systems for cyber security. In: *3th International Conference on System of Systems Engineering (SoSE)*, 2011, p. 13–18.

Walraven, S.; Lagaisse, B.; Truyen, E.; Joosen, W. Dynamic composition of cross-organizational features in distributed software systems. In: *10th International Conference on Distributed Applications and Interoperable Systems (DAIS)*, 2010, p. 183–197.

Wang, H.; Wan, C. Quality failure prediction for the self-healing of service-oriented system of systems. In: *21th International Conference on Web Services (ICWS)*, 2014, p. 566–573.

Xia, X.; Wu, J.; Liu, C.; Xu, L. A Model-Driven Approach for Evaluating System of Systems. In: *18th International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2013, p. 56–64.

Zafar, N.; Arnautovic, E.; Diabat, A.; Svetinovic, D. System security requirements analysis: A smart grid case study. *Systems Engineering*, v. 17, n. 1, p. 77–88, 2014.

Zhu, L.; Staples, M.; Jeffery, R. Scaling up software architecture evaluation processes. In: *9th International Conference on Software Process (ICSP)*, 2008, p. 112–122.

Zuccato, A. b.; Daniels, N.; Jampathom, C.; Nilson, M. Report: Modular safeguards to create holistic security requirement specifications for system of systems. In: *2th International conference on Engineering Secure Software and Systems (ESSoS)*, 2010, p. 218–230.

Chapter

A

Quality Attributes for Systems-of-Systems: Systematic Literature Review

In order to conduct this SLR, this work followed the process proposed by Kitchenham and Charters (2007). In short, this process presents three main phases: (i) Planning: in this phase, the research objectives and the systematic review protocol are defined. The protocol constitutes a pre-determined plan that describes research questions and how the systematic review will be conducted; (ii) Conduction: during this phase, primary studies are identified, selected, and evaluated according to the inclusion and exclusion criteria previously established. For each selected study, data are extracted and synthesized; and (iii) Reporting: in this phase, a final report is organized and presented. As the results of this SLR were already reported in Chapter 2, this appendix describes the first and second phases in more details.

A.1 Planning

To establish the systematic review protocol it is necessary to states that the main research objective of this systematic review is to identify primary studies that address quality attributes for SoS considering its software domains. Thus, aiming at finding possibly all primary studies to understand and summarize evidences about quality attributes in SoS, the following research questions (RQ) were established:

- **RQ1**: What are the most common quality attributes for SoS?;
- **RQ2**: Which are the most common application domains considered for SoS?; and
- **RQ3**: Which are the QA established for each SoS domains?

In order to establish the search strategy, it is defined the following elements:

Sources selection criteria: Three types of sources were considered in this systematic review: (i) Publication databases: Aiming at establishing which publication database would be used to find the primary studies, it was adopted the following criteria (Dieste et al., 2009): *content update* (publications are regularly updated); *availability* (full text of the papers are available); *quality of results* (accuracy of the results obtained by the search); and *versatility export* (a mechanism to export the results of the search is available); (ii) Related works: it was also considered those studies cited as related works in the relevant primary studies found in the publication databases; and (iii) Specialist: it was also considered studies suggested by specialists of the areas of system of systems and software quality. Although the indication of studies by specialists can be considered as bias, it was adopted this source aiming to not lose any important evidence.

Sources list: The publication databases selected for this systematic review are shown in Table A.1. According to Dybå and Dingsøyr (2008), these databases are efficient to conduct systematic reviews in the context of Software Engineering. Furthermore, *Scopus* has been added, since it is considered the largest database of abstracts and citations (Kitchenham and Charters, 2007);

Table A.1: Selected publication databases

Source	**Location**
ACM Digital Library	`http://dl.acm.org/`
IEEE Xplore	`http://ieeexplore.ieee.org/Xplore/home.jsp`
Scopus	`http://www.scopus.com/home.url`
Web of Science	`http://apps.isiknowledge.com/`

Studies language: Only primary studies written in English were considered in this systematic review. English was adopted because most of research in Computer Science have been reported in this language;

Keywords: The main keywords were "*System-of-Systems*" and "*Quality Attributes*", with the following related terms: (i) **System-of-Systems**: system-of-systems, systems of systems, and systems-of-systems; and (ii) **Quality Attributes**: quality attribute, quality attributes, non functional requirement, non-functional requirement, nonfunctional requirement, non functional requirements, non-functional requirements, nonfunctional requirements, quality requirement, quality requirements, quality characteristic, quality characteristics, quality criteria, quality criterion OR non functional property, non-functional property, nonfunctional property, non functional properties, non-functional properties, nonfunctional properties, non functional characteristic, non-functional characteristic, non-functional characteristic, non functional characteristics, non-functional characteristics, nonfunctional characteristics, quality models, and quality model.

Search string: It was applied the boolean OR operator to link the main terms and their related terms. All these terms were combined using the boolean AND operator. Thus, the final search string was:

(“system of systems” OR “system-of-systems” OR “systems of systems” OR “systems-of-systems”) AND (“quality attribute” OR “quality attributes” OR “non functional requirement” OR “non-functional requirement” OR “nonfunctional requirement” OR “non functional requirements” OR “non-functional requirements” OR “nonfunctional requirements” OR “quality requirement” OR “quality requirements” OR “quality characteristic” OR “quality characteristics” OR “quality criteria” OR “quality criterion” OR “non functional property” OR “non-functional property” OR “nonfunctional property” OR “non functional properties” OR “non-functional properties” OR “nonfunctional properties” OR “non functional characteristic” OR “non-functional characteristic” OR “nonfunctional characteristic” OR “non functional characteristics” OR “non-functional characteristics” OR “nonfunctional characteristics” OR “quality models” OR “quality model”)

Another important activity of the systematic review planning is to define the Inclusion Criteria (IC) and Exclusion Criteria (EC). These criteria make possible to include primary studies that are relevant to answer the research questions and exclude studies that do not answer them. Thus, the inclusion criteria are:

- **IC1**: The study address one quality attribute for SoS;
- **IC2**: The study address more than one QA for SoS;
- **IC3**: The study address QA for SoS in one application domain; and
- **IC4**: The study address QA for SoS in more than one application domain.

The exclusion criteria are:

- **EC1**: The study does not address SoS;
- **EC2**: The study does not propose QA for SoS;

- **EC3**: The study is an editorial, keynote, opinion, tutorial, poster or panel;
- **EC4**: The study is a previous version of a more complete study about the same research;
- **EC5**: The paper language is different from English;
- **EC6**: The paper is duplicated; and
- **EC6**: The full paper is not available.

In this systematic review, the selection and evaluation of primary studies was be performed in three steps:

1. **First Selection:** initially, the search string will be customized and applied to each publication databases, previously listed in Table A.1. For this, the title, abstract, and keywords of all primary studies available will be considered. As a result, a set of primary studies possibly related to the research topic will be obtained. Based on this set, the title and the abstract of each primary study will be read and the inclusion and exclusion criteria will be applied. Thus, studies will be selected as relevant or not. The introduction and the conclusion sections of each primary study might also be considered when necessary. After this analysis, if a study has been selected, it will be read in full;
2. **Second Selection:** in this step, each primary study selected will be read in full and analyzed again according to inclusion and exclusion criteria. In addition, the related works cited by these studies will be evaluated intending to cover the whole research area. This additional selection might be a great source of evidence, since an included study often presents related works in the same research area. If the decision about the inclusion or exclusion of a study is not clear, this study will be analyzed by two reviewers. When a disagreement occurs, discussions will be conducted; and
3. **Data Extraction and Synthesis Method:** in order to extract data, it was planned to build data extraction tables related to each research question. These tables should synthesize results to facilitate the drawing of conclusions. To summarize and describe the set of data, statistical synthesis method and meta-analysis was applied.

A.2 Conduction

This SLR was conducted by three people: a software engineering researcher, a senior software developer/architect from the industry, and a systematic review specialist. The work started in October/2014 and was finalized in December/2014. In this section, the details of the procedures to select the studies are described.

The primary studies were firstly identified on the selected databases, according to the systematic review planning previously established. As a result, 116 studies were identified. Next, the primary studies were selected by reading of title and abstract and by the application of the inclusion and exclusion criteria. The Table A.2 summarizes the number of studies obtained on each database, the number of included studies, and the precision rate ACM, IEEE Xplore, Scopus, and Web of Science showed precision rates of 0.173, 0.181, 0.073, and 0.875, respectively. It is important to highlight that mainly Scopus stores studies from other databases, such as IEEE Xplore and ACM. After that, repeated studies were also excluded and a total of 50 studies were selected. Then, the inclusion and exclusion criteria were again applied after full reading of the selected studies. Finally, 40 studies were selected as the most relevant to this systematic review. At the end, 12 studies were included by the reviewers that were not returned originally by the selected databases due to another systematic review performed by the authors of this work (Santos et al., 2014) accounting a total of 52 studies included.

Table A.2: Search sources and primary studies selected and included

Source	**Selected**	**Included**	**Precision**
ACM Digital Library	5	5	1
IEEE Xplore	14	9	0.643
Scopus	89	42	0.472
Web of Science	8	7	0.875

Table A.3 shows the 52 primary studies included. This table also presents the citation for each study and the inclusion criteria (IC) used to include it. Column "Document Type" indicates if the primary study was published as a Journal Article (JA) or as a Conference Paper (CP).

Table A.3: Included primary studies

#	**Citation**	**IC**	**Type**
S1	Ackermann, C., Lindvall, M. and Cleaveland, R. Towards Behavioral Reflexion Models.(Ackermann et al., 2009)	IC2,IC4	CP
S2	Alghamdi, A.; Hussain, T. and Khan, G. Enhancing C4I security using threat modeling.(Alghamdi et al., 2010)	IC2,IC3	CP
S3	Allen, M. From substandard to successful software. (Allen, 2009)	IC2,IC3	JA
S4	Aoyama, M. and Tanabe, H. A Design Methodology for Real-Time Distributed Software Architecture Based on the Behavioral Properties and Its Application to Advanced Automotive Software. (Aoyama and Tanabe, 2011)	IC2,IC3	CP

continued on next page ...

#	Citation	IC	Type
S5	Azizian, N., Mazzuchi, T., Sarkani, S. and Rico, D. A framework for evaluating technology readiness, system quality, and program performance of U.S. DoD acquisitions. (Azizian et al., 2011)	IC2,IC3	JA
S6	Babar, M.A.a and Chauhan, M.A.b. A tale of migration to cloud computing for sharing experiences and observations. (Babar and Chauhan, 2011)	IC2,IC3	CP
S7	Balci, O. and Arthur, J. D. and Ormsby, W. F. Achieving reusability and composability with a simulation conceptual model. (Balci et al., 2011)	IC2,IC4	JA
S8	Batista, T. Challenges for sos architecture description. (Batista, 2013)	IC2,IC4	CP
S9	Belloir, Nicolas and Chiprianov, Vanea and Ahmad, Manzoor and Munier, Manuel and Gallon, Laurent and Bruel, Jean-Michel. Using Relax Operators into an MDE Security Requirement Elicitation Process for Systems of Systems. (Belloir et al., 2007)	IC2,IC4	CP
S10	Calinescu, R.a and Kikuchi, S.b and Johnson, K.a. Compositional reverification of probabilistic safety properties for large-scale complex IT systems. (Calinescu et al., 2012)	IC2,IC3	JA
S11	Hong-Mei Chen and Kazman, R. Architecting ultra-large-scale green information systems. (Chen et al., 2012)	IC1,IC3	CP
S12	Chiprianov, Vanea and Falkner, Katrina and Gallon, Laurent and Munier, Manuel. Towards modelling and analysing non-functional properties of systems of systems. (Chiprianov et al., 2014)	IC2,IC4	CP
S13	Colin, C.V. and Duncan, J.R. and Lu, L. and Zongyang, L. and David, E.W. and Jie, X. A scenario-based architecture evaluation framework for network enabled capability. (Venters et al., 2009)	IC2,IC3	CP
S14	Eklund, U.a c and Bosch, J.b. Architecture for embedded open software ecosystems. (Eklund and Bosch, 2014)	IC2,IC3	JA
S15	Fang, Z., DeLaurentis, D. and Davendralingam, N. An Approach to Facilitate Decision Making on Architecture Evolution Strategies. (Fang et al., 2013)	IC2,IC3	JA
S16	Fischer, D., Sarkarati, M., Spada, M., Michelbach, T., Urban, W. and Tueffers, C. An application security framework for SOA-based mission data systems. (Fischer et al., 2011)	IC1,IC3	CP
S17	Fuchs, A. and Rieke, R. Identification of security requirements in systems of systems by functional security analysis. (Fuchs and Rieke, 2010)	IC2,IC3	JA
S18	Gagliardi, M., Wood, W., Klein, J. and Morley, J. A uniform approach for system of systems architecture evaluation. (Gagliardi et al., 2009)	IC2,IC3	JA
S19	Garro, A. and Tundis, A. On the Reliability Analysis of Systems and SoS: The RAMSAS Method and Related Extensions. (Garro and Tundis, 2014)	IC1,IC4	JA

continued on next page ...

#	Citation	IC	Type
S20	Ge, B., Hipel, K. W., Yang, K. and Chen, Y. A data-centric capability-focused approach for system-of-systems architecture modeling and analysis. (Ge et al., 2013)	IC2,IC3	JA
S21	Guariniello, C. and DeLaurentis, D. Communications, Information, and Cyber Security in Systems-of-Systems: Assessing the Impact of Attacks through Interdependency Analysis. (Guariniello and DeLaurentis, 2014a)	IC2,IC3	JA
S22	Guariniello, C. and DeLaurentis, D. Integrated Analysis of Functional and Developmental Interdependencies to Quantify and Trade-off Ilities for System-of-Systems Design, Architecture, and Evolution. (Guariniello and DeLaurentis, 2014b)	IC2,IC4	CP/JA
S23	Guo, F. and Wang, M. Quantitative measurement of interoperability by using Petri net. (Guo and Wang, 2012)	IC1,IC3	JA
S24	Iacobucci, J. and Mavris, D. A method for the generation and evaluation of architecture alternatives on the cloud. (Iacobucci and Mavris, 2011)	IC1,IC3	CP
S25	Joyce, H., Hin, O., Seng, Y. and Chan, S. A systems assurance perspective towards generic systems engineering. (Joyce et al., 2011)	IC1,IC3	CP
S26	Kazman, R. b., Gagliardi, M. and Wood, W. Scaling up software architecture analysis. (Kazman et al., 2012)	IC2,IC3	JA
S27	Kimura, D., Osaki, T., Yanoo, K., Izukura, S., Sakaki, H. and Kobayashi, A. Evaluation of IT systems considering characteristics as system of systems. (Kimura et al., 2011)	IC2,IC3	CP
S28	Leuchter, S., Reinert, F. and Muller, W. Assessment of the integration capability of system architectures from a complex and distributed software systems perspective. (Leuchter et al., 2014)	IC2,IC3	CP
S29	Lin, Q., Cai, Z., Wang, Y., Yang, J. and Chen, L. Adaptive flight control design for quadrotor UAV based on dynamic inversion and neural networks. (Lin et al., 2013)	IC2,IC3	CP
S30	Madni, A. and Sievers, M. System of systems integration: Key considerations and challenges. (Madni and Sievers, 2014)	IC1,IC3	JA
S31	Michael, J., Riehle, R. and Shing, M.T. The verification and validation of software architecture for systems of systems. (Michael et al., 2009)	IC2,IC4	CP
S32	Nair, S., De La Vara, J., Sabetzadeh, M. and Briand, L. An extended systematic literature review on provision of evidence for safety certification. (Nair et al., 2014)	IC1,IC3	JA
S33	Oliveira, M. and Pereira, J. Extensible Virtual Environment Systems Using System of Systems Engineering Approach. (Oliveira and Pereira, 2007)	IC2,IC3	CP

continued on next page ...

#	Citation	IC	Type
S34	Osmundson, J. and Langford, G. Connections in system of systems. (Osmundson and Langford, 2012)	IC1,IC4	CP
S35	Ramaswamy, A. b., Monsuez, B. and Tapus, A. Formal models for cognitive systems. (Ramaswamy et al., 2013)	IC2,IC3	CP
S36	Rezaei, R., Chiew, T. and Lee, S. An interoperability model for ultra large scale systems. (Rezaei et al., 2014)	IC1,IC4	JA
S37	Schneider, D., Becker, M. and Trapp, M. Approaching runtime trust assurance in open adaptive systems. (?)	IC2,IC3	CP
S38	Schneider, D. and Trapp, M. A safety engineering framework for open adaptive systems. (Schneider and Trapp, 2011)	IC1,IC3	CP
S39	Schneider, D. and Trapp, M. Runtime safety models in open systems of systems. (Schneider and Trapp, 2009)	IC2,IC3	CP
S40	Schugerl, P., Rilling, J., Witte, R. and Charland, P. A quality perspective of software evolvability using semantic analysis. (Schugerl et al., 2009)	IC2,IC4	CP
S41	Shukla, M. and Asundi, J. Considering emergency and disaster management systems from a software architecture perspective. (Shukla and Asundi, 2012)	IC2,IC3	JA
S42	Singh, A. and Dagli, C. H. Multi-objective stochastic heuristic methodology for tradespace exploration of a network centric system of systems. (Singh and Dagli, 2009)	IC2,IC3	CP
S43	Stratton, W., Sibol, D., Lindvall, M., Ackermann, C. and Godfrey, S. Developing an approach for analyzing and verifying system communication. (Stratton et al., 2009a)	IC2,IC3	CP
S44	Tsadimas, A., Kapos, G.-D., Dalakas, V., Nikolaidou, M. and Anagnostopoulos, D. Integrating simulation capabilities into SysML for enterprise information system design. (Tsadimas et al., 2014)	IC1,IC3	CP
S45	Tucker, A. and Dagli, C. Design of experiments as a means of lean value delivery to the flight test enterprise. (Tucker and Dagli, 2009)	IC2,IC3	JA
S46	Urwin, E., Venters, C., Russell, D., Liu, L., Luo, Z., Webster, D., Henshaw, M. and Xu, J. Scenario-based design and evaluation for capability. (Urwin et al., 2010)	IC2,IC3	CP
S47	Wada, H., Suzuki, J. and Oba, K. A model-driven development framework for non-functional aspects in service oriented architecture. (Wada et al., 2008)	IC2,IC4	JA
S48	Walraven, S., Lagaisse, B., Truyen, E. and Joosen, W. Dynamic Composition of Cross-organizational Features in Distributed Software Systems. (Walraven et al., 2010)	IC2,IC4	CP
S49	Xia, X., Wu, J., Liu, C. and Xu, L. A Model-Driven Approach for Evaluating System of Systems. (Xia et al., 2013)	IC2,IC3	CP
S50	Zafar, N., Arnautovic, E., Diabat, A. and Svetinovic, D. System Security Requirements Analysis: A Smart Grid Case Study. (Zafar et al., 2014)	IC1,IC3	JA

continued on next page ...

#	Citation	IC	Type
S51	Zhu, L., Staples, M. and Jeffery, R. Scaling up software architecture evaluation processes. (Zhu et al., 2008)	IC2,IC4	JA
S52	Zuccato, A. b., Daniels, N., Jampathom, C. and Nilson, M. Report: Modular safeguards to create holistic security requirement specifications for system of systems. (Zuccato et al., 2010)	IC1,IC4	JA

A.3 Threats to Validity

The main threats identified to the validity of this SLR are described as follows:

- **Missing of important primary studies:** the search for studies related to SoS quality attributes was conducted in several publication databases and search engines. According to Dybå and Dingsøyr (2008) and Kitchenham and Charters (2007), the publication databases that were used are the most relevant sources. Aiming at not missing any important evidence, it was also considered the specialist suggestion and the related works which were presented in the reference list of the selected primary studies. In addition, no limit was placed on the date of publications. During the search, conference papers, journals, and technical reports were also considered. In spite of the effort to included all relevant evidence in this research, it is possible that primary studies were missed;

- **Reviewers reliability:** in this SLR, all reviewers are researchers in the Software Engineering area. Furthermore, none of primary study was published by research group or researchers related to the authors of this work. Therefore, the authors of this work are not aware of any bias that may have been introduced during the analysis process. However, it might be possible that the conclusion about the studies evaluated have been influenced by the opinion of the reviewers;

- **Data extraction:** another threat to this review refers to how the data were extracted from the primary studies, since not all the information were obvious to answer the research questions and some data had to be interpreted. In order to ensure the validity of this systematic review, other sources of information were analyzed, i.e., technical reports, web sites, and manuals, in addition to the primary studies analyzed. Furthermore, in the event of a disagreement between the reviewers, a discussion was conducted to ensure that a full agreement was reached; and

- **Quality assessment:** since the goal of this systematic review was to identify studies related to SoS quality attributes, no quality assessment was performed, as it might restrict the number of primary studies included. It is an agreement that a quality assessment can provide more insights and explanations to the conclusion of this review. Thus, it will be included in a future version of this work.

As it can be observed, the authors of this work are concerned with the validity of results of this systematic review. In particular, they have dedicated special effort to completely cover this research area as impartial as possible.

Chapter

B

Evaluation of Systems-of-Systems Software Architectures: Systematic Literature Review

In order to conduct this systematic review, this work followed the process proposed by Kitchenham and Charters (2007). In short, this process presents three main phases: (i) Planning: in this phase, the research objectives and the systematic review protocol are defined. The protocol constitutes a predetermined plan that describes research questions and how the systematic review will be conducted; (ii) Conduction: during this phase, primary studies are identified, selected, and evaluated according to the inclusion and exclusion criteria previously established. For each selected study, data are extracted and synthesized; and (iii) Reporting: in this phase, a final report is organized and presented.

B.1 Planning

The main goal of this SLR is to understand how the evaluation of the SoS software architecture has been performed and which architecture evaluation methods have been used or proposed for this purpose. In this sense, it is very important to understand how the quality requirements has been achieved through architectural evaluation and which are the impacts on the quality of SoS as a whole, considering their different application domains. Finally, aiming to contribute to the creation of new research perspectives, this SLR intend to identify the difficulties and challenges of this application context. Thus, aiming at finding possibly all primary studies to understand and summarize evidences about SoS software architecture evaluation, the following research questions (RQ) were established:

- **RQ1:** How the SoS architecture have been evaluated?
- **RQ2:** What is the maturity level of SoS architecture evaluation approaches?
- **RQ3:** How the architecture evaluation contributes for the fulfillment of SoS quality requirements?
- **RQ4:** What are the difficulties and challenges in the SoS architecture evaluation?

The search string was defined based on the analysis of the research questions and calibrated through of a control group consisting of three relevant studies in the research area: (Gagliardi et al., 2009; Kazman et al., 2012; Xia et al., 2013).

Thus, an initial search string was tested in some of the predefined search bases and improved until all articles present in the control group were returned. Furthermore, the keywords to compose the generic search string were selected to cover the largest number of studies regarding SoS architecture evaluation. Thereafter, the generic search string was created from the key words: “System-of-Systems”, “Software Architecture” and “Evaluation” plus their respective synonyms through the use of boolean operators AND and OR. Thus, the final search string was:

(“system of systems” OR “system-of-systems” OR “systems of systems” OR “systems-of-systems”) AND (“software architecture” OR “architectural” OR “architecting” OR “software architectures” OR “nonfunctional requirement”) AND (“assessment” OR “verification” OR “validation” OR “validate” OR “evaluation” OR “evaluate” OR “analysis” OR “analyze” OR “validating” OR “analyzing” OR “evaluating”)

In order to covering the largest number of studies related to SoS architecture evaluation, it was used automatic search in publication databases instead of a limited set of journals and conferences. The bases used are shown in Table B.1. It is noteworthy that from the generic string were derived specific strings for each database.

Another important activity of the systematic review planning is to define the Inclusion Criteria (IC) and Exclusion Criteria (EC). These criteria are used to identify primary studies that provide direct evidence on the research questions. For this SLR, the following criteria were defined:

Table B.1: Selected publication databases

Source	**Location**
ACM Digital Library	`http://dl.acm.org/`
IEEE Xplore	`http://ieeexplore.ieee.org/Xplore/home.jsp`
Scopus	`http://www.scopus.com/home.url`
Web of Science	`http://webofknowledge.com`
ScienceDirect	`http://sciencedirect.com`

The inclusion criteria are:

- **IC1:** Studies that presents some level of evaluation or architectural analysis for SoS, regardless of the application domain; and
- **IC2:** Studies reporting difficulties or challenges identified in the SoS architecture evaluations.

The exclusion criteria are:

- **EC1:** Duplicated studies;
- **EC2:** Studies not written in English; and
- **EC3:** Studies that do not present or describe any architectural evaluation initiative in the SoS context.

B.2 Conduction

This SLR was conducted by six people: three software engineering researchers, and three systematic review specialists. The work started in May/2014 and was finalized in July/2014. In this section, the details of the procedures to select the studies are described.

Thereafter the search string calibrated, the search in the publication databases was performed. The process of identification of primary studies is presented in the Figure B.1. Initially, it were identified 292 primary studies. However, the research in selected bases resulted in overlapping studies, which required the exclusion of duplicates through manual filtering. In this step, remained 176 studies, which were analyzed by reading of title and abstract and by the application of the inclusion and exclusion criteria. The divergences between the reviewers were discussed and resolved during this phase. Finally, it were selected 16 primary studies, which were fully read and submitted to the extraction of relevant information. Still at this stage, 1 study was excluded because do not presents an SoS evaluation in the architectural level. Therefore, 16 primary studies were selected to answer the research questions. The list of selected studies and the main information extracted are presented respectively in the Table B.2 and Table B.3.

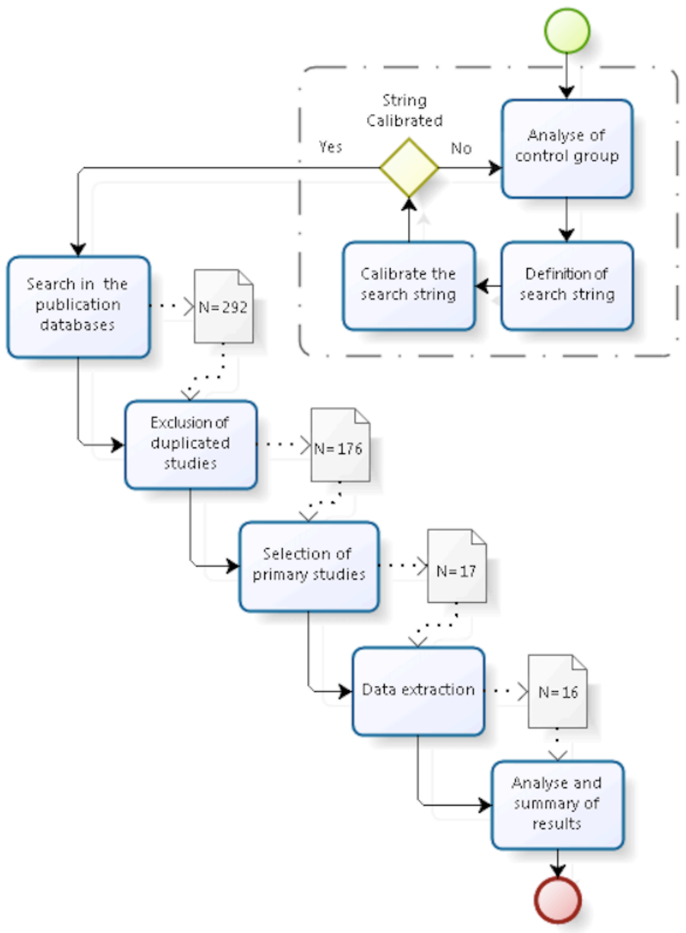

String Calibrated
Yes
No
Analyse of control group
Definition of search string
Calibrate the search string
Search in the publication databases
N=292
Exclusion of duplicated studies
N=176
Selection of primary studies
N=17
Data extraction
N=16
Analyse and summary of results

Figure B.1: Process of primary studies selection

Table B.2: Primary studies selected

ID	Author	Name	Year
E1	Guarinielloa, C. and DeLaurentisa, D	Communications, information, and cyber security in systems-of-systems: Assessing the impact of attacks through interdependency analysis	2014
E2	Guarinielloa, C. and DeLaurentisa, D	Integrated analysis of functional and developmental interdependencies to quantify and trade-off ilities for system-of-systems design, architecture, and evolution	2014
E3	Fanga, Z., DeLaurentisa, D. and Davendralingam N.	An Approach to Facilitate Decision Making on Architecture Evolution Strategies	2013
E4	Ge, B., Hipel, K. W., Yang, K. and Chen, Y.	A data-centric capability-focused approach for system-of-systems architecture modeling and analysis	2013
E5	Xia, X., Wu J., Liu C. and Xu L.	A Model-Driven Approach for Evaluating System of Systems	2013
E6	Chen, H. and Kazman, R	Architecting ultra-large-scale green information systems	2012
E7	Kazman, R., Gagliardi, M. and Wood, W.	Scaling up software architecture analysis	2012
E8	Iacobucci. J. and Mavris, D.	A method for the generation and evaluation of architecture alternatives on the cloud	2011
E9	Urwin, E., Venters, C., Russell, D., Liu, L., Luo, Z., Webster, D., Henshaw, M. and Xu, J.	Scenario-based design and evaluation for capability	2010
E10	Ackermann, C., Lindvall, M. and Cleaveland, R.	Towards behavioral reflexion models	2009
E11	Gagliardi, M., Wood, W., Klein, J. and Morle, J.	A uniform approach for system of systems architecture evaluation	2009
E12	Michael, J. B., Riehle, R. and Shing, M.	The verification and validation of software architecture for systems of systems	2009
E13	Singh, A. and Dagli, C. H.	Multi-objective stochastic heuristic methodology for tradespace exploration of a network centric system of systems	2009
E14	Stratton, W. C., Sibol, D. E., Lindvall, M., Ackermann, C. and Godfrey, S.	Developing an approach for analyzing and verifying system communication	2009
E15	Zhu, L., Staples, M. and Jeffery, R.	Scaling up software architecture evaluation processes	2008
E16	Oliveira, M. and Pereira, J.	Extensible virtual environment systems using system of systems engineering approach	2007

Table B.3: Summary of SoS architecture evaluation approaches

ID	Evaluation Approach	Evaluation Phase	Methods and Techniques	Validation	Quality Attributes	Domain
E1	Mathematical modeling	Maintenance phase	FDNA	Proof of concept	Reliability, Capability, Operability, Security	Military
E2	Mathematical modeling	Both phase	FDNA, DDNA	Proof of concept	Operability, Flexibility	Generic
E3	Mathematical modeling	Maintenance phase	CPN	Case study	Performance, Complexity	Military
E4	Simulation-based	Maintenance phase	CPN	Proof of concept	Generic	Generic
E5	Simulation-based	Both phase	-	Case study	Performance, Effectiveness	Military
E6	Scenario-based	Design phase	ATAM, CBAM	Industry	Generic	Electric power
E7	Scenario-based	Both phase	ATAM, QAW	Industry	Generic	Military
E8	Simulation-based	Design phase	-	Case study	Performance	Military
E9	Scenario-based	Both phase	AEF	Expert opinion	Performance, Survivability, Lethality	Military
E10	Mathematical modeling	Both phase	-	Case study	Conformity	Generic
E11	Scenario-based	Both phase	ATAM, QAW	Proof of concept	Generic	Military
E12	Mathematical modeling	Both phase	-	No assessment	Generic	Generic
E13	Mathematical modeling	Design phase	GA, FIS	Proof of concept	Maintainability, Reliability, Flexibility, Complexity, Affordability	Electric power
E14	Simulation-based	Maintenance phase	-	Case study	Reliability	Generic
E15	Scenario-based	Design phase	ATAM, CBAM	Industry	Generic	Industry
E16	Scenario-based	Both phase	SAAM	Expert opinion	Generic	Virtual environment

B.3 Threats to Validity

The main threats identified to the validity of this SLR are described as follows:

- **Missing of important primary studies:** the search for studies related to SoS architecture evaluation was conducted in several publication databases and search engines. According to Dybå and Dingsøyr (2008) and Kitchenham and Charters (2007), the publication databases that were used are the most relevant sources. In addition, no limit was placed on the date of publications. During the search, conference papers, journals, and technical reports were also considered. In spite of the effort to include all relevant evidence in this research, it is possible that primary studies were missed;
- **Reviewers reliability:** in this SLR, all reviewers are researchers in the Software Engineering area. Furthermore, none of primary study was published by research group or researchers related to the authors of this work. Therefore, the authors of this work are not aware of any bias that may have been introduced during the analysis process. However, it might be possible that the conclusion about the studies evaluated have been influenced by the opinion of the reviewers;
- **Data extraction:** another threat to this review refers to how the data were extracted from the primary studies, since not all the information were obvious to answer the research questions and some data had to be interpreted. Furthermore, in the event of a disagreement between the reviewers, a discussion was conducted to ensure that a full agreement was reached; and
- **Quality assessment:** since the goal of this systematic review was to identify studies related to SoS architecture evaluation no quality assessment was performed, as it might restrict the number of primary studies included. It is an agreement that a quality assessment can provide more insights and explanations to the conclusion of this review. Thus, it will be included in a future version of this work.

As it can be observed, the authors of this work are concerned with the validity of results of this systematic review. In particular, they have dedicated special effort to completely cover this research area as impartial as possible.

B.4 Main Results

Although there are several architecture evaluation methods, the scenario-based methods such as ATAM (Kazman et al., 2000), and SAAM (Kazman et al., 1994), are the most popular ones and the most used in the industrial context (Babar et al., 2004; Mårtensson and Tekniska, 2006; Nakagawa et al., 2013). However, these methods still have several limitations that difficult their application in the SoS context, mainly because they rely on resources that are not usually available in this context. One of the most critical limitations is that the reliability of the software architecture evaluation is directly related the evaluators knowledge in the evaluation method applied, which makes the evaluation very subjective, and the heavy involvement of stakeholders in the evaluation process which greatly increases the cost of the evaluation due to the difficulty of meeting a large numbers of people (Barcelos and Travassos, 2006). These resources are usually unavailable in the SoS, since the architectural documents are created by different development teams, from different tools and notations, where there are several stakeholders, geographically distributed, with competing interests. This scenario makes difficult to focus on quality attributes and increases the level of complexity, cost and effort required in architecture evaluation process (Gagliardi et al., 2009).

Aiming to provide an overview about current approaches for evaluating the SoS architecture, a Systematic Literature Review (SLR) was carried out following the guidelines established by Kitchenham and Charters (2007). The focus is to understand how the evaluation of the SoS architecture has been performed, which methods and techniques have been used and how the SoS quality requirements have been met through architecture evaluation. In addition, we surveyed the difficulties and challenges that are inherent to SoS evaluation as a way of pointing out new and important research perspectives in the software architecture area. Overall, 16 primary studies were included in this SLR. The next section presents the main results obtained in this SLR considering the scope of this Master's project. The research protocol (i.e., the process followed for identifying, analyzing, and extracting information from primary studies) and details about the execution of this SLR are presented in Appendix A. In addition, part of these results can be found in (Santos et al., 2014).

B.4.1 Evaluation Methods and Techniques

Evaluation of software architectures usually occurs after the design of such architectures but before implementation starts. Nonetheless, an architecture can be evaluated at any stage of its life cycle (Clements et al., 2002). In particular, for SoS software architectures, due to their characteristics, we have observed that most works have proposed application of evaluation methods in the design phase, as well as in architectures already established, intending to analyze their flexibility and ability to evolution.

The processes for evaluation of SoS software architectures are typically supported by different methods and techniques. Usually, they are adapted and enhanced to create a new proposal for SoS software architectures. Overall, six of the sixteen works propose or use mathematical-modeling evaluation methods, four works propose or use simulation-based methods, and six works propose or use scenario-based methods. Among the scenario-based approaches, ATAM (Kazman et al., 2000) is the most popular one. In our SLR, it was not observed a convergence in using a specific type of evaluation method. Besides that, we did not found works that consider experience-based methods for evaluating SoS.

The suitability and viability of the methods and techniques proposed usually have been assessed through expert opinion and experiences in real projects, proof of concept or demonstration, case study, and application in industry. It is important to highlight that all works selected to evaluation in the industrial context proposed the use of scenario-based evaluation methods. This result shows that these evaluation methods have been well accepted by industry or at least could be scalable to SoS.

B.4.2 Quality Attributes Commonly Considered

Evaluation methods can either focus on single or several quality attributes. Our results showed that all proposed scenarios-based methods do not focus on specific quality attributes. The main reason is that scenarios-based methods usually focus on identifying trade-off among different quality attributes instead of measuring each one. However, simulation-based and mathematical modeling methods usually focus on one or a few tangible quality attributes. The most common quality attributes considered by these methods are reliability, performance, operability, complexity, and flexibility. However, we have observed that evaluation methods for SoS should take into account several quality attributes. Moreover, these methods should also be able to measure and classify these quality attributes to support an accurate comparison among architectural alternatives. This may be possible through the use of simulation-based in combination with scenarios-based approaches.

Finally, the use of quality models for evaluating SoS architectures would be relevant, as they would provide standardization for quality attributes of SoS, as well as establishment of relationships among such attributes. However, none of the works included in our SLR discusses the use of quality models during architectural evaluation.

B.4.3 Evaluation Challenges

We have also observed that there are several challenges for an adequate evaluation of SoS software architectures. The following discussion focuses on the main challenges.

The reliability of the communication among constituent systems is an important factor to the success of SoS (Urwin et al., 2010). According to Stratton et al. (2009b), it is difficult to ensure reliable communication through an architecture evaluation for several reasons: (i) constituent systems are usually developed independently by different teams at different places; (ii) specification of communication requirements is ambiguous; and (iii) communication issues are often subtle and remain hidden for a long time. Moreover, the complex interdependencies that exist among constituents make it difficult to foresee the behavior of SoS due to an unexpected loss of one of their constituents. In the worst case, SoS could collapse or trigger a cascading failure among their constituents. These consequences cannot be fully understood through an architectural evaluation of the independent systems, as SoS require an evaluation of the effect of interdependence among constituents on the entire system (Guariniello and DeLaurentis, 2014a).

Regarding the evolutionary and decentralized nature of SoS, it becomes difficult to ensure, for instance, reliability, security, or performance, using architecture evaluation methods, which focus exclusively on structural characteristics but ignore behavior compliance. This can be a problem, as a simple divergence in the implementation of one of the constituents often reduces performance and reliability of the entire SoS (Ackermann et al., 2009; Chen et al., 2012; Zhu et al., 2008).

Finally, an important step to an adequate architectural evaluation involves the identification of metrics to measure features of systems. However, metrics used to evaluate individual systems can not directly deal with the characteristics of the SoS (Guariniello and DeLaurentis, 2014b). This happens because the emergent behavior of SoS usually cannot be captured and evaluated by evaluation approaches that address the level of constituent systems (Meilich, 2006).

In summary, the source of challenges and difficulties related to SoS architectures is mostly due to its emergent behavior that can not be captured and evaluated only through classical software engineering. The context of SoS difficult the establishment of limits to the specification and system design to create a set of requirements to be evaluated (Meilich, 2006).

B.5 Conclusion

Despite the number of initiatives to evaluate SoS software architectures found in the literature, there is still no consensus on what exactly should be considered during this evaluation. From our results, we observe that main challenges in the SoS architecture evaluation are due to the complex interaction among constituent systems and the evolutionary, distributed nature of SoS as well. Therefore, appropriate, scalable evaluation approaches still need to be developed. Moreover, we envisage that these new approaches should be able to successfully capture and evaluate the emergent behavior of SoS.

As future work, we intend to continue our investigation on evaluation of SoS architectures, updating this SLR as well as identifying appropriate architecture evaluation methods that consider quality attributes usually addressed by SoS. Moreover, we will investigate alternatives to combine these methods and techniques to reduce the number of difficulties and challenges that are inherent to this new class of complex, large software systems.