A quality model for critical embedded systems

Brauner Roberto do Nascimento Oliveira
A quality model for critical embedded 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. FINAL VERSION

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Elisa Yumi Nakagawa

USP – São Carlos
July 2017
A quality model for critical embedded systems / Brauner Roberto do Nascimento Oliveira; orientador Elisa Yumi Nakagawa. -- São Carlos, 2017. 145 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. critical embedded systems. 2. software architecture. 3. software quality. I. Nakagawa, Elisa Yumi, orient. II. Título.
Brauner Roberto do Nascimento Oliveira

Um modelo de qualidade para sistemas embarcados críticos

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. VERSÃO REVISADA
Área de Concentração: Ciências de Computação e Matemática Computacional
Orientadora: Profa. Dra. Elisa Yumi Nakagawa

USP – São Carlos
Julho de 2017
Dedico este trabalho em memória de meu pai,
Braulino Mendonça de Oliveira, e minha irmã,
Flávia Roberto do Nascimento Oliveira,
que com tão pouco tempo vivendo ao meu lado,
me fazem amá-los eternamente em minha jornada.
As pessoas são as coisas mais importantes deste mundo. Não importa se fazem o bem ou o mau. Cabe a nós, antes de mais nada, compreendê-las e compartilhar de nossas vidas com elas. Só assim é possível sonhar com um mundo repleto de amor e coisas boas. É deste modo que começo esta seção tão importante de meu trabalho, pois eternamente agradecido sou por cada uma das pessoas que estiveram e estarão presentes durante minha trajetória.

O maior dos amores não pode ser escrito nem dito. Trata-se de um sentimento recíproco, verdadeiro e infinito que nos motiva em todos os momentos, sejam eles bons ou ruins. Não importa a distância e nem mesmo o tempo. Por esse amor, serei eternamente grato à minha mãe, Osania Roberto do Nascimento Oliveira, que me fez chegar até aqui.

Agradeço imensamente a Prof. Elisa Yumi Nakagawa por me orientar na condução deste trabalho. Sua dedicação e envolvimento foram fundamentais para meu desenvolvimento pessoal e profissional.

À Lina Maria Garcés-Rodriguez, minha eterna gratidão pelo apoio e incentivo ao longo dos últimos três anos.

À minha família, pelo respeito, consideração e confiança depositados em mim.

À Profa. Silvia Maria do Espírito Santo, pela confiança e incentivo para ingressar na pós-graduação.

Ao Prof. Dr. Adenilso da Silva Simão, pelo respeito à minha atuação enquanto representante discente e consideração pelos alunos do PPG-CCMC.

À todos os professores que tive a oportunidade de encontrar desde o ensino básico, pois me fizeram admirar essa profissão de mudar o mundo.

Aos amigos da PuraTI, Diego e André, pela oportunidade de trabalhar juntos por um objetivo e pelo exemplo de perseverança.

À todos (e são muitos) colegas e amigos do LabES, pelas inúmeras experiências, aprendizados e lições.

Ao serviço de pós-graduação do ICMC, em especial à Carol Murata, Leonardo Miyake e Lycaena Couvre.

Ao CNPq, CAPES e FAPESP pelo apoio financeiro.
“Será algo de maravilhoso quando vislumbrarmos a verdadeira personalidade do homem. Crescerá naturalmente, simplesmente, à maneira das flores ou das árvores. Nunca se porá em discórdia, nem entrará em discussões ou contendas. Nada terá de provar. Conhecerá tudo. E no entanto não se ocupará do conhecimento. Será sábia. Bens materiais não medirão seu valor. Não haverá de ter coisa alguma. E terá no entanto todas as coisas; tão rica, o que dela venha a se tirar, ela ainda o haverá de ter. Não estará sempre se intrometendo com os demais, ou pedindo-lhes para serem iguais a si própria. Ela os amará por serem diferentes. E embora não vá se intrometer com os demais, ajudará a todos, como algo de belo nos ajuda por ser o que é.”

(Oscar Wilde)
RESUMO

OLIVEIRA, B. R. N. Um modelo de qualidade para sistemas embarcados críticos. 2017. 145 p. Master dissertation (Master student Program in Computer Science and Computational Mathematics) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

Sistemas embarcados, incluindo sistemas embarcados críticos (SEC), estão cada vez mais presentes na sociedade moderna, provendo soluções específicas que variam de sistemas pequenos até sistemas grandes e complexos, como é possível encontrar em carros, aviões e equipamentos médicos. Falhas nesses sistemas podem resultar em danos à seres humanos e ao meio ambiente, ou então em uma perda financeira irrecuperável. Sendo assim, é muito importante garantir que os SEC sejam construídos e apresentem um nível adequado de qualidade. Para garantir que esses sistemas alcancem tal qualidade, é necessário considerar sua arquitetura de software, já que esta impacta de maneira significativa na qualidade do software enquanto artefato ou produto. Deste modo, o objetivo deste projeto de mestrado é de propor um modelo de qualidade que contém os atributos de qualidade mais importantes para SEC, servindo como artefato para apoiar a execução de atividades arquiteturais (além de outras que possam se beneficiar) tais como análise e avaliação, no contexto de SEC.

Palavras-chave: Sistemas embarcados críticos, arquitetura de software, qualidade de software.
ABSTRACT

OLIVEIRA, B. R. N. A quality model for critical embedded systems. 2017. 145 p. Master dissertation (Master student Program in Computer Science and Computational Mathematics) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

Embedded systems, including critical embedded systems (CES) are increasingly present in the modern society, providing specific solutions from small to large complex systems, such as in cars, airplanes, and healthcare equipment. Failures in these systems can result in damage to human beings, and to the environment, or can represent an unrecoverable financial loss. In this sense, it is very important to ensure they are built with quality. To provide such quality, it is necessary to consider their software architecture, which impacts on the quality of the systems. This way, the main objective of this Master’s project is to propose a quality model concerning the most important quality attributes for CES, which may be used to support (but not limited to) architectural activities such as analysis and evaluation in the context of CES.

Keywords: Critical embedded systems, software architecture, software quality.
| Figure  | Description                                                                                           | Page |
|--------|-------------------------------------------------------------------------------------------------------|------|
| 1      | Embedded Systems Architecture                                                                         | 28   |
| 2      | Hardware in the loop                                                                                  | 28   |
| 3      | Memory size vs access time/energy consumption                                                         | 30   |
| 4      | V-model (Development Process)                                                                         | 32   |
| 5      | Embedded technology into an automobile                                                                | 33   |
| 6      | Architectural design activities                                                                        | 39   |
| 7      | ATAM Concept interactions                                                                             | 43   |
| 8      | DCAR main steps (left column) and artifacts generated throughout the process (right column)           | 46   |
| 9      | Graphical view on trade-offs between performance, security and dependability (BARBACCI et al., 1995)    | 49   |
| 10     | Reliability and validity of measures                                                                  | 54   |
| 11     | DAP Classification for Q-Models                                                                       | 55   |
| 12     | The quality model for classifying and evaluating automotive embedded software architectures             | 62   |
| 13     | Embedded software component Quality Model (EQM)                                                       | 63   |
| 14     | Quality model for embedded system                                                                     | 63   |
| 15     | Process followed to conduct the Systematic Mapping                                                    | 65   |
| 16     | Procedure performed for Quality Attribute mapping                                                     | 70   |
| 17     | Frequency of Quality Characteristics                                                                  | 71   |
| 18     | Overall process followed to develop the quality model                                                 | 78   |
| 19     | CES concept map                                                                                        | 80   |
| 20     | Survey Process                                                                                        | 98   |
| 21     | Boxplot of time spend to complete each survey question group and total time. Black dots are outliers   | 106  |
|        | (two are omitted for better visualization)                                                            |      |
| 22     | Boxplot of each QA regarding score achieved in likert-scale                                           | 108  |
| 23     | Boxplot of each QA regarding score achieved in likert-scale                                           | 109  |
| 24     | Boxplot of each QA regarding score achieved in likert-scale                                           | 110  |
| 25     | Correlation matrix for QAs scores. It is showing only correlation coefficients between QAs that       | 112  |
|        | achieved a significance level of 0.05 (p-value)                                                       |      |
Figure 26 – Relations submitted by participants. The number labelling the lines represents the number of times the relation was submitted. The color of the lines and their thickness is based on their frequency: green (2 times), blue (3 times), golden (4 times), and red (7 times).

Figure 27 – Relations submitted by participants. The number labelling the lines represents the number of times the relation was submitted. The color of the lines and their thickness is based on their frequency: green (2 times), blue (3-7 times), golden (8 times), and red (18 times).
| Table | Description |
|-------|-------------|
| 1     | Quality attribute scenario for Modifiability (BASS; CLEMENTS; KAZMAN, 2012) | 49 |
| 2     | List of all studies selected for extraction activity | 65 |
| 3     | Information summary for RQ1 | 69 |
| 4     | Quality attributes organized by domain | 72 |
| 5     | QAs of interest in many application domains | 75 |
| 6     | Product quality characteristics from ISO/IEC 25010, definitions, and selection flag | 81 |
| 7     | Quality in use characteristics from ISO/IEC 25010, definitions, and selection flag | 85 |
| 8     | Final version of QM4CES | 95 |
| 9     | Goals, questions and metrics for survey | 99 |
| 10    | Participant information summary | 104 |
| 11    | Descriptive statistics for time spent (in minutes) in survey, divided by time spent in each group and total survey time | 106 |
| 12    | Descriptive statistics for QAs: minimum value, 1st quartile, median, standard deviation ($\sigma$), coefficient of variation ($c_v$), mean, mode, 3rd quartile, and max value | 107 |
| 13    | Relative frequency of values for each QA in likert-scale | 111 |
| 14    | Computed values from reliability analysis | 113 |
| 15    | Mean, median and mode for DUMOD scale supplied for new attributes | 117 |
# CONTENTS

1 INTRODUCTION .................................................. 21
1.1 Contextualization ........................................... 21
1.2 Motivation ................................................... 22
1.3 Objective ..................................................... 23
1.4 Organization ................................................ 23

2 BACKGROUND .................................................. 25
2.1 Critical Embedded Systems ................................. 25
2.1.1 Definition ............................................... 26
2.1.2 Characteristics of CES ................................. 27
2.1.3 Development Process of ES ............................ 31
2.1.4 Application Domains .................................... 33
2.1.5 Challenges ............................................... 35
2.2 Software Architecture ..................................... 36
2.2.1 Brief History ........................................... 37
2.2.2 Software Architecture and System Qualities .... 38
2.2.3 Architectural Process .................................. 38
2.2.4 Architectural Evaluation ............................... 42
2.3 Software Quality ........................................... 47
2.3.1 Quality Attributes ...................................... 47
2.3.2 Quality Metrics ......................................... 52
2.3.3 Quality Models .......................................... 53
2.3.4 Quality Model Establishment ......................... 56
2.3.5 Domain-Oriented Quality Models Development (DUMOD) 56
2.4 Final Remarks ................................................ 60

3 QUALITY IN THE CONTEXT OF CRITICAL EMBEDDED SYS-
TEMS ............................................................... 61
3.1 Related Work ................................................ 61
3.2 Systematic Literature Mapping ......................... 64
3.2.1 Conduction .............................................. 64
3.2.2 Reporting the mapping results ...................... 64
3.2.3 Discussion of results .................................. 75
1.1 Contextualization

Embedded systems (ESs) are an important type of software-intensive systems, as they have been widely used in the modern society as part of the people’s lives (MARWEDEL, 2011; AGUIAR et al., 2010; SILVA; VIEIRA, 2013; RODRIGUES, 2011) (e.g., mobile phones, cars, fabrication and healthcare equipment, avionic, transportation, military, and global communication systems). Some of them are considered critical (i.e., Critical Embedded Systems - CESs) as their failure may cause serious damage to human lives, environment and expensive equipment, and unrecoverable financial losses as well (RODRIGUES, 2011; MARWEDEL, 2011; ANDALAM et al., 2013; ARMOUCH; SALEWSKI; KOWALEWSKI, 2008). Therefore, CESs present a number of quality requirements to avoid or mitigate any kind of critical damage (MARWEDEL, 2011). These requirements are translated into quality attributes (QA), i.e., properties that the system must exhibit, such as reliability, security, and safety (OLIVEIRA et al., 2013). An important aspect of QA is that they can be measured using software quality metrics, allowing engineers to determine the degree to which a QA is present in a system. For example, in order to measure the learnability of a system, a metric is the time of training required before independent use of the system by users, calculated in hours for different users in different versions of the same product (FENTON; PFLEEGGER, 1998).

In order to support the achievement of QA requirements, quality models (QM) have been consolidated. In short, QM present a definition and rationale for QA, categorizing them in such a way that it is possible to identify relations among QA, through a hierarchical and taxonomic approach (ISO/IEC/IEEE 24765., 2010). Moreover, QM have become well-accepted by the community to describe, manage, and predict software quality (DEISSENBOECK et al., 2009). Many QM have been proposed since the emergence of Software Engineering, for example McCall Quality Model (1977) (MCCALL; RICHARDS; WALTERS, 1977), Boehm’s Quality Model (1976)(BOEHM; BROWN; LIPOW, 1976), Dromey’s Quality Model (1995) (DROMEY,
Chapter 1. Introduction

In parallel, the software architecture of a system plays a fundamental role in the realization of QA requirements (BASS; CLEMENTS; KAZMAN, 2012). In short, Software architecture is defined as the structure of a system that should be designed in early phases of the project to reduce efforts in later phases, such as implementation and maintenance (BASS; CLEMENTS; KAZMAN, 2012). In the context of software architecture, two activities plays an important role to its life-cycle: architectural analysis and evaluation. During architectural analysis, engineers focus on gathering QA requirements or architectural significant requirements (ASR) from requirements documents, if available, system stakeholders through interviews, and/or business goals (BASS; CLEMENTS; KAZMAN, 2012). This is the first step before the design of the software architecture. Thereafter, to ensure that the architecture will enable the final product to exhibit its QA and fulfill the requirements, architectural evaluations can be driven during the architecture design or after its completion, by designers and/or peers related to its design process or by specialized companies (BASS; CLEMENTS; KAZMAN, 2012). Architecture evaluations have many benefits, however, the most important one is to identify problems and risks as soon as possible, for easily and less-costly fixing such problems and risks during architecture design or posterior system life-cycle phases (HEESCH et al., 2014). In this scenario, QM are commonly used to support the specification of software architecture from different perspectives, including architecture analysis and evaluation activities, and, therefore, play an important role when considering trade-offs among different architectural solutions (ISO/IEC 25010, 2011).

1.2 Motivation

Most of the time, CES have been analyzed, developed, tested, and deployed without considering strictly QA, what can be impacting their quality. Moreover, although many software QM have been proposed (e.g., (BOEHM; BROWN; LIPOW, 1976; MCCALL; RICHARDS; WALTERS, 1977; ISO/IEC 25010, 2011; DROMEY, 1995; WAGNER et al., 2012), they are considered as general-purpose and are not focused on critical QA of CES. The number of QA of these models can make it hard to apply them for specific application domains.

Furthermore, three QM that could be used in the context of CES were found in a systematic literature review reported by (OLIVEIRA et al., 2013). (AHRENS et al., 2011) established a specialized QM to conduct the evaluation of software architectures in the context of driver assistant systems. (CARVALHO; MEIRA, 2009) proposed a framework, which includes a QM, to support the verification of embedded software components. (JEONG; KIM, 2011) established a QM of lightweight components for embedded components. Only the first study developed a QM aiming at the conduction of an architectural activity, but it deals only with automotive domain. The latter two studies proposes QM from the component-based software
1.3 Objective

In this context, the main goal of this project was to systematically establish a QM for CES concerning the most relevant QA that have been addressed in literature. Such a model can be mainly used to support architectural activities performed in the context of CES. In this sense, we expected to contribute to the quality of activities performed during CES development and, by consequence, improve the quality of CES.

1.4 Organization

This dissertation is organized as follows. Chapter 2 presents background of critical embedded systems, software architecture and software quality. A systematic literature mapping regarding software quality (models, attributes and metrics) in the context of CES is presented in Chapter 3. Chapter 4 presents the initial steps followed to establish the preliminary quality model and its final structure resulted from the steps performed in Chapter 5. Chapter 5 presents the final step for model establishment, addressing a survey conducted with practitioners and researchers with experience in CES and the analysis of results as predicted by the DUMOD approach. Finally, Chapter 6 presents the conclusions of the author about the development of this work.

engineering (CBSE) perspective and even they might be applied to CES domain, they were not developed exclusively for such systems. In all of these studies, no systematic approach was followed to establish QM, as result, they might be incomplete or miss important elements.
In this chapter the main areas related to this project are presented. We start presenting definition, characteristics (hardware and software components), development process, application domains, and challenges of critical embedded systems (CES). After that, we address the concept of software architecture, highlight its role in achieving system qualities, and the processes of architectural analysis and evaluation. Finally, software quality is presented from the point-of-view of quality attributes (QA), quality metrics, and quality models (QM). Moreover we discuss the development of QM with emphasis on a particular approach that is employed in this work.

### 2.1 Critical Embedded Systems

Embedded systems (ES) command and control operations through an integration of complex hardware components and software, and are usually characterized by strict requirements for time-to-market, fault tolerance, and robustness (DOBRICA; NIEMELÄ, 2008). They can be found in our daily lives, like in healthcare electronics, automotive industry (e.g., cruise control systems) or communication devices (e.g., in cell phones), playing a fundamental role in a recently introduced type of computing: ubiquitous or pervasive computing, where the main goal is to make information available anytime, anywhere (MARWEDEL, 2011). ES are often implemented based on micro-controllers or Field Programmable Gate Array (FPGA) (SADDEM; TOGUYENI; TAGINA, 2011) and are a key and primary source of leverage for innovation (GILL, 2005).

Some of these systems are considered critical (i.e., CES) as they have tight constraints about safe operation (GILL, 2005) and the absence of errors (JANUZAJ et al., 2010). Furthermore, in case of misbehavior, they need to quickly respond to themselves in a safe working mode (SADDEM; TOGUYENI; TAGINA, 2011). Our daily lives depend more and more on CES, from communications to transportation and medicine. Software is embedded in many high-tech products and increasingly defines the innovation and market capability of a range of industrial
sectors. A CES combines various high-tech hardware and software elements. Many of these systems have considerable economic value and strict quality requirements, such as dependability, security, safety, and performance. Software elements in a software-intensive system are often as large as millions of lines of code and determine any of the following points: system functionality, system cost, system development risk, and development time. They are present in several application areas, such as:

**Automotive electronics:** Technologically advanced cars include air bag control systems, engine control systems, Anti-Braking Systems (ABS), Electronic Stability Programs (ESP) and other safety features, air-conditioning, GPS-systems, anti-theft protection, and more. In addition, autonomous vehicles have been a research subject, as human driving errors are a major cause of car accidents on roads (FERNANDES et al., 2012), making dependability and safety examples of the most important system requirements;

**Avionics:** Flight control systems, anti-collision systems, pilot information systems, and others compose a significant amount of the total value of airplanes and are extremely important to the dependability that airplanes must offer. Security, maintenance, safety, availability, integrity, performance, and others are examples of non-functional requirements that are found in the avionics application domain (PAULITSCH; RUSS; SOREA, 2008); and

**Health sector:** Healthcare products play an important role in healthcare, where an improvement to such products has a direct impact on the medical service. Many non-functional requirements must be achieved in products of the health sector to avoid any kind of damage to human beings. For example, reliability, safety, functionality, portability, modifiability, testability, and serviceability are considered the most important quality attributes throughout the development of a medical imaging product family (WIJNSTRA, 2001).

Besides these areas, some other applications areas in CES domain are: railways, telecommunication, fabrication equipment, smart buildings, logistics, and robotics.

In the next sections ES and CES are discussed in more detail.

### 2.1.1 Definition

CES are a special class of ES in which design errors can potentially be catastrophic (BATE, 2008), in terms of causing serious damage to the environment or to human lives, or non-recoverable material and financial losses (AGUIAR et al., 2010); therefore, they are distinguished by their rigorous requirements that must be achieved before they are deployed into their target production environment. Additionally, to understand what is a CES, it’s also necessary to define what is an ES. However, there is no unique definition. This is due to the fact that the evolution of these systems follows the advances in technology and decreases in the
2.1. Critical Embedded Systems

cost of both hardware and software components used for their implementation (NOERGAARD, 2012). Despite this fact, they can be defined as information processing systems embedded into enclosing products (MARWEDEL, 2011). Additionally, ES are commonly defined by their comparison against personal computers (PCs). For many ES representatives, the functionality limitation of hardware and/or software is greater than for PCs (NOERGAARD, 2012). In terms of hardware, limitation means less processing performance, power available, memory and hardware functionality. In turn, software limitations mean fewer and/or scaled-down applications, limited or no operating systems, and lower abstraction-level code. But, as said before, the growing advances in technology is changing this scenario. The availability of more memory for the same or less previously occupied physical space allows more applications and data to be allocated for a system, increasing its capabilities. Faster processing units reduces time taken for data processing, and so on. Still, keeping the comparison with PCs, ES are generally designed to perform a dedicated function, in contrast to the generic purpose of PCs. However, in the recent years, this is becoming less true since the appearance of devices, such as personal data assistants (PDA), cellphones, and smart phones, which are capable of performing a variety of primary functions (NOERGAARD, 2012). However, some authors state that they evolved to an extend that they can not be considered ES anymore, even though they share many characteristics.

In the next section we discuss characteristics of hardware and software of these systems.

2.1.2 Characteristics of CES

ES command and control operations through an integration of complex hardware components and software, and are usually characterized by strict requirements for time-to-market, fault tolerance, and robustness (DOBrica; NIEMELÄ, 2008). The understanding of the hardware platform in which a ES is being developed is extremely important for all system designers (NOERGAARD, 2012). In this sense, artifacts created by hardware engineers to describe how developed hardware works must be understandable even by programmers (NOERGAARD, 2012). Hardware descriptions contain the necessary information to support a suitable embedded software development concerning its compatibility with the hardware platform, and provide insights on how to communicate hardware requirements of software to a hardware engineer (NOERGAARD, 2012). In turn, embedded software is responsible to provide, in most systems, an interface in which the hardware is controlled (NOERGAARD, 2012). In this case, they are called device drivers and stand right in the middle of hardware and system software layer, providing a common interface in which hardware communicates with the operating system, middleware, and the upper application layer (NOERGAARD, 2012). Figure 1 shows three common architectures regarding the composition of ES.

In the following section, it is presented some of the most common characteristics of these systems accordingly to their composition from both hardware and software perspectives.

Hardware Components
Hardware components are used in a loop on ES, as shown in Figure 2. In this loop, the system acts on a physical environment and is capable of sense and control it. This is the common model for control systems, but can be employed as a first order approximation for other applications (MARWEDEL, 2011).

From the loop model, hardware components are classified into five groups (MARWEDEL, 2011): input; processing units; memories; communication; and output. Following, these groups are further explained.

Input hardware is responsible for providing the very first data necessary for ES to perform their main function. Sensors, sample-and-hold circuits, and A/D (analogical to digital) converters are examples of input hardware. Sensors gather information about the environment and can be designed for virtually every physical quantity, including weight, velocity, acceleration, electrical current, voltage, temperature, etc. (MARWEDEL, 2011). A huge amount of them have been designed on the last years, supporting the development of many smart systems (MARWEDEL, 2011). Examples include rain sensors used on cars to control the speed of wipers based on the amount of rain, therefore removing distraction from drivers, biometric sensors support high security standards and are already very common on current systems, and radio frequency identification (RFID) technology, which consists of an integrated circuit and an antenna that responds to radio frequency, able of being embedded into objects and live beings for identification purposes. On the other hand, sample-and-hold circuits convert signals obtained through sensors.
over the continuous domain into the discrete domain where digital computers work. Finally, A/D converters replace signals that map time to continuous value domain by those that map time to a discrete value domain, which is a must when considering digital computers.

Processing units are the main functional hardware responsible for processing system instructions, and data provided by input hardware (MARWEDEL, 2011). These units are classified into three main groups (MARWEDEL, 2011): (i) Application Specific Integrated Circuit (ASIC); (ii) processors; and (iii) reconfigurable logic. The main differences between them concerns their efficiency (e.g., processing power vs energy consumption), flexibility (e.g., how hard is to change the software system), and cost. Following, some characteristics of each group are show.

ASICs are developed to fulfill the requirements of applications with very tight constraints on performance, efficiency and size. For that reason, they are commonly adopted when maximum energy efficiency is needed or if the market accepts the costs or the amount of products that can be sold is sufficient enough to overcome the investment (MARWEDEL, 2011). Additionally, the design of ES with ASICs suffers from long iteration times and lacks of flexibility, therefore, design errors lead to the acquisition of new masks used to transfer data into the chip. This results in a new fabrication run, increasing the development costs (MARWEDEL, 2011).

Processors, in turn, exhibit flexibility as their main advantage. ES built using processors can have their overall behavior changed just by changing the software that runs on the processor (MARWEDEL, 2011). Therefore, this allows designers to easily correct design errors and update the system to perform new functionalities (MARWEDEL, 2011). However, the popularity of processors comes with a trade-off. More flexibility means less efficiency. Systems that aims at excellent flexibility can not be power efficient (MARWEDEL, 2011). In despite of this fact, there are many available techniques that can make processors more energy efficient.

Reconfigurable logic becomes an alternative when ASICs can not be employed because of market constraints and software-based solutions running on processors are either too slow to satisfy real-time constraints or too energy consuming (MARWEDEL, 2011). Reconfigurable hardware is commonly implemented using RAM memory to store run-time configurations during normal operation, and uses ROM and Flash memories to store configuration data that will be loaded to RAM on system’s start up. The most common form of reconfigurable hardware are the Field Programmable Gate Arrays (FPGAs). These devices can be programmable after fabrication.

The adoption of memories for ES is based on system requirements concerning efficiency and performance. For this, the memory hierarchy can be explored in order to fill the system’s requirements. In general, the main problem for ES is that larger memories imply in more energy and time per access, as shown in Figure 3.

Communication hardware allows the transference of information among system components through various channels. A channel is an abstract entity characterized by communication properties such as information transfer capacity and noise parameters (MARWEDEL, 2011).
In turn, the physical entity that enables communication is called communication media, and can be classified as wireless media (e.g., radio frequency and infrared), optical media (e.g., optical fibers), and wires (MARWEDEL, 2011). The selection of communication media is impacted by system requirements such as real-time behavior, efficiency, communication delay, support for event-driven communication, robustness, fault tolerance, maintainability, and privacy (MARWEDEL, 2011). Examples of this kind of hardware includes sensor/actuator buses, which provides communication among simple devices such as switches or lamps and the processing units, field buses, which are similar to sensor/actuator buses but support larger data transfer rates, wired multimedia communication supports even more larger data transfer rates, and, lastly, wireless communication which is very common nowadays with the emergence of technologies such as Bluetooth, and wi-fi.

Last but not least, output hardware for ES includes displays, electro-mechanical devices, D/A-converters, and actuators (MARWEDEL, 2011). Displays output information about the operations performed by the system to the user that will make use of it. Examples includes organic (MARWEDEL, 2011) and LCD displays. Electro-mechanical devices interact with the environment, influencing it through motors and other equipment. If analog devices are used, then, it is necessary to convert the digital information into analog, through the use of D/A converters (MARWEDEL, 2011). Lastly, actuators are devices that converts signals to their corresponding physical action like moving tons of weight (MARWEDEL, 2011).

**Software Components**

Most ES requires some kind of software to perform their initialization and management. Software is present in both system and software application layers. The system software layer comprises the device driver layer and middleware or operating system layer. In turn, application layer is characterized by software built upon the system software layer. This layer is very important because the system being built is characterized by the functionalities performed by applications embedded into the product, i.e., the application layer represents at the highest level the purpose of the ES and does most of the interactions with users (NOERGAARD, 2012).
Embedded operating systems support the development of more complex applications which requires process, memory and I/O management (MARWEDEL, 2011). Therefore, they are an abstraction layer built on top of hardware components making the development of applications easier and less dependent on hardware (MARWEDEL, 2011). However, differently of general purpose OSs used in PCs, these systems are limited by strict constraints and requirements found on the ES domain. Therefore, an embedded OS must be small and can not be completely flexible (i.e., provide every development library) towards application development. For systems that requires real-time computations, real-time operating systems (RTOS) can be employed. These systems must have their timing behavior predicable in various levels, manage time, manage the scheduling of tasks, and last but not least, be fast (MARWEDEL, 2011). Although some RTOSs have been designed for multi-purpose embedded applications, some have focused on specific domains. This is the case of operating systems compliant to the OSEK/VDX standard for automotive control (MARWEDEL, 2011). As a result, these RTOSs can provide a dedicated service for that domain, being more compact and efficient. The Enterprise Real-time Kernel (ERIKA) is one example.

Middleware is software that mediates communication between the application software and the kernel or device driver software. It is an abstraction layer generally used on embedded devices that have two or more applications, in order to provide some flexibility, security, portability, and other quality characteristics to the communication among applications (MARWEDEL, 2011). They can be independent of the OS and are useful to reduce the complexity and redundancy of applications that would embed additional code for portability purposes. In contrast, the use of middleware adds overhead to the system, impacting, in some cases, on the scalability and performance of systems (MARWEDEL, 2011). Message Oriented Middleware (MOM), Object Request Brokers (ORBs), and Remote Procedure Calls (RPCs) are types of middleware elements (NOERGAARD, 2012).

2.1.3 Development Process of ES

As many other systems, the development process of ES may be dictated by models presented by both disciplines of system and software engineering. In general, these models present many similarities concerning which activities should be carried, but vary on how and when the tasks are performed, and if they are repeatable throughout the development process. The major difference between the process to develop information systems and ES is the addition of hardware concerns to the design phase, including at least a hardware selection phase, the hardware/software partitioning problem, and hardware/software integration. A very common model presented by the ES literature for system development is shown in Figure 4.

The V-model is a variation of the very first well-know waterfall model for software development. This model depicts the relationship of quality assurance activities to the actions associated to communication, modeling, and early implementation activities (PRESSMAN,
As a development team moves down the left side of the V, initial requirements are refined into progressively more detailed and technical representations of the problem and its solution (PRESSMAN, 2010). Once the implementation code has been generated, the team moves up the right side of the V, essentially performing a series of tests that validate each of the models created as the team moved down the left side. In general, this life-cycle model provides a way to visualize how verification and validation actions are applied during earlier system engineering (PRESSMAN, 2010).

The specification and modeling of ES must be able to reflect and support, accordingly to the criticality of the application being developed, the following requirements (MARWEDEL, 2011): hierarchy (behavioral and structural) in many ways; component-based design; concurrency; synchronization and communication; timing-behavior; stated-oriented behavior; event-handling; exception-oriented behavior; presence of programming elements; executability; design of large systems; domain-specific requirements; readability; portability and flexibility; termination; support for non-standard I/O-devices; non-functional properties; support for the design of dependable systems; no obstacles to the generation of efficient implementations; and an appropriate model of computing (MoC).

During early design phases, the concepts that will drive the system requirements are commonly specified in a informal way, since they probably gonna be changed during design iterations. For this, the potential uses of the system under design (SUD) can be captured in form of use cases, which can be specified in different ways, including UML diagrams. In a slightly more detailed level, some might want to explicitly model the sequence in which messages must be exchanged between system components. This may be done using UML system sequence diagrams for previously defined use cases. A variant of this diagram is the time/distance diagram (TDD), which models time in addition to sequence (MARWEDEL, 2011). In a more detailed level, some other models can be employed. Communicating finite state machines (CFSMs) are
used to model the system behavior in a state-oriented approach. To fulfill the requirements about time, classical automata are extended to Timed automata. However, describing SUDs using state machine diagrams is very hard and can not express complex computations (MARWEDEL, 2011). Although standard programming languages may be used to express complex computations, the execution of several threads on a multi-threaded environment may be unpredictable, which is unacceptable for hard real-time constraints. In this sense, synchronous languages merge both finite state machines and programming languages in one model, enabling the expression of complex computations based on a finite automata execution model (MARWEDEL, 2011). Examples of these languages include Esterel, Lustre, and SCADE.

2.1.4 Application Domains

As stated before, ES are found in many areas. In the following sections are presented some details and examples of three areas: automotive electronics, avionics, and railways.

Automotive Electronics

Nowadays, modern cars can be sold in technologically advanced countries only if they contain a significant amount of embedded electronics systems (MARWEDEL, 2011). These include air bag control systems, engine control systems, anti-braking systems (ABS), electronic stability programs (ESP) and other safety features, air-conditioning, GPS-systems, anti-theft protection, and many more. Figure 5 shows some of them along with their sensors, connected through a Controller Area Network (CAN) bus.

Figure 5 – Embedded technology into an automobile ((AAICAR, 2014))

Anti-lock Braking Systems (ABS) have been widely used in automotive industry because of their addition to safety properties of vehicles such as cars, trucks, and buses (CHAO; JIANMIN;
ABS, as the name suggests, are employed to replace mechanical braking systems to prevent a vehicle’s wheel from locking as a result of excessive operation of the service brake by the driver, especially on slippery road surfaces (CHAO; JIANMIN; YONGCHUAN, 2010). This way, the lateral control of the vehicle is maintained even on extreme situations that requires the driver to press the brake pedal to its maximum intensity, even for long periods of time. With the vehicle under certain control, the driver can avoid crashes that were impossible to avoid with locked wheels. Additionally, ABS optimize the utilization of the available adhesion coefficient between tires and the road surface and thus increase vehicle retardation and decrease stopping distance (LEE et al., 2002).

Another safety device that is growing popular is the air-bag system. Nowadays, it is possible to find many applications for air-bags, varying from automobiles to helicopters. For automotive industry, air-bags are used to protect automobile occupants from accidental crashes. For this, there are some types of air-bags that are activated to protect different parts of the body, including side, front, knee, and seat belt air-bags. An air bag system is composed of four essential components (CHAN, 2002): (i) sensors that have the main role of detecting and measuring an impacting; (ii) an inflator that is triggered by the sensors to generate gases to fill the bag; (iii) an bag that when inflated provides a cushion to absorb the kinetic energy of the occupant; and (iv) vehicle interior that accommodates the installation of the air bag system.

Avionics

Many ES can be found on airplanes. As previously listed, flight control systems, anti-collision systems, pilot information systems, and others do compose a lot of the airplane overall value, and perform critical and specific functions to help pilots to maintain the airplane flying safely. Sometimes, systems in this domain are called ultra-critical ES (PAULITSCH; RUESS; SOREA, 2008), this is due to the presence of airplanes in the commercial and military context, used to transport a lot of peoples and cargo across different regions of the world, being very important to the many levels of market and national defense.

Accordingly to (PAULITSCH; RUESS; SOREA, 2008), there is a trend in avionics to use Integrated Modular Architectures (IMAs). In this architecture, subsystems are integrated into a common avionics platform characterized by a general compute platform and I/O interfaces. I/O devices and computers are connected by a shared system network. This architecture seems to impacts positively on scalability and obsolesce. This trend is responsible for drawing the attention for non-functional requirements such as security, maintainability, dependability, performance, testability, and schedulability.

Railways

For railways, the applicability of ES is similar to the situation discussed for cars and airplanes (MARWEDEL, 2011). As an example, a Railway Interlocking System (RIS) can be explored. An RIS is a CES which ensures that it is impossible to manually or automatically (i.e.,
by other systems) set switch points of the train way that may lead to train collision (DIPOPPA et al., 2001). The development of this kind of system must ensure its correctness, raising its production costs and time-to-market. Their safety is also regulated by standards such as the ENELEC50128/129 (CENELEC-50128., 2000; CENELEC-50129., 1999).

Another type of CES used in this domain is the Automatic Train Protection (ATP). This system is the core subsystem of Communication Based Train Control (CBTC), which is a control and protection system for trains that includes functions such as train speed determination, over speed protection and brake assurance, rollback protection, emergency braking and so on (IEEE 1474.1., 2004). The main responsibility of ATP is to protect the system from over speed (XIONG et al., 2010). It is composed of the following subsystems (which are devices, too) (XIONG et al., 2010): (i) Processing Unit that is is responsible for receiving commands from train lines and compare the train speed with limiting speed, and emits commands to the braking equipment to brake the train; (ii) Train lines trough which commands and limiting speed are sent to Processing Unit; (iii) Mode Direction Handler which enables the driver to select the drive mode. (iv) Speed Sensor that sends current train speed to Processing Unit; (v) Train Operation Display which shows commands and speed information from Train lines and warnings from Processing Unit; and (vi) Train Information Management for the management of train information such as drive mode and corresponding limiting speed.

2.1.5 Challenges

Additionally to the inherent characteristics of ES previously showed, challenges to develop CES are emerging since their massive employment in many contexts, begin part of the day-to-day of peoples, and the rapid growth of hardware and software technologies. While in the past the problem was considerable simple by programming small and resource-constrained electronic devices, nowadays the problem is the span of a new generation of complex, distributed, and real-time systems (GILL, 2005). This lead to some discussions about themes like ubiquitous computing, pervasive computing, ambient intelligence, the disappearing computer, and the post-PC era (MARWEDEL, 2011).

Some new challenges and requirements that may be expected are (GILL, 2005):

- The increased demand for system functionalities also increases the expectation for these systems to operate largely autonomously while performing with almost instant response times. This may be the case of automatic activation of braking systems such as ABS;
- Performance-based designs are needed in many areas constrained by efficiency and usage-problems. Examples vary from the continuous control of hybrid gas-electric automobile engines to the complex control of ultra-tech air vehicles that must accommodate extreme situations such as vertical takeoff and landing;
• Strict closed designs belongs to the past, that is, attention is being draw by open systems, which are capable of interacting with equally IT-intensive contexts, sharing computational resources. This is the case of the Integrated Modular Avionics (IMA) cited as an example for embedded domains. Approaches like the IMA is responsible for both computing platform cost-savings and the performance gains of integration among systems, and drawn attention for interoperability concerns;

• Past complex systems were built using centralized, distributed designs. Increasingly, networked systems were aggregated to compose other systems, rather then primary designed to be distributed. This is the case of plug and play approaches used in new concepts of medical intensive care units, developed to accommodate situational usage of oxygen, temperature, blood glucose sensors, infusion pumps, ventilators, and surgical micro-robotic devices; and

• Intelligent, self-healing, and adaptive systems are already focus of research. So we may expect more capable systems with embedded learning and adaptive technologies. In turn, this also refers to a well understanding of such technologies, rigorous design approaches to ensure safe operation, fault tolerance and isolation, security from unauthorized access, and autonomous recovery.

2.2 Software Architecture

Software architecture can be defined as the set of structures needed to reason about the system, comprising software elements, relations among them, and properties of both (BASS; CLEMENTS; KAZMAN, 2012). Many other definitions have been given to software architecture, but, in general, there is an agreement that software architecture deals with the macroscopic and sweeping issues of the software design (FAIRBANKS, 2010). A software architecture acts as the skeleton abstraction of the system, and, therefore, is present in any system, even if it was not designed during early phases of the software project. (BASS; CLEMENTS; KAZMAN, 2012) list thirteen reasons they consider the most important to justify the importance of the software architecture discipline. Some of these reasons are: (i) a software architecture inhibit or enable system’s quality attributes; (ii) it allows developers to reason about the system during its evolution; (iii) if documented, it enhances the communication among stakeholders; (iv) the architecture carries the early and most fundamental, hardest-to-change design decisions; and (v) it allows system qualities prediction through early architectural evaluations.

During the design of the system’s software architecture, the activities of architectural analysis and evaluation play an important role to the successful deployment of the architecture and therefore the remainder activities of the software development. In the architectural analysis, engineers focus on the specification of architecturally significant requirements (ASR). These requirements can be gathered from the system’s requirements document, through stakeholders
software architecture (BASS; CLEMENTS; KAZMAN, 2012). A system requirement is considered an ASR when it has a great impact on the architecture design (e.g., quality attribute requirements). During this activity, some challenges emerge (NIU et al., 2014): (i) while it is fairly easy to gather explicit ASRs, it is very hard to gather implicit ones; (ii) during the ASRs modeling, it is hard to reason about the trade-offs among the interrelated and sometimes competing needs; and (iii) communication among stakeholders is affected by terminological problems, e.g., when stakeholders use different terminology for the same concepts, same terminology for different concepts, and/or differ in terminology and concepts.

In order to identify problems and risks early on, architectural evaluations can be conducted by comparing the requirements established during architectural analysis to architectural solutions which were proposed to satisfy them. In addition, it is possible evaluate the architecture of existing systems in preparation for maintenance or upgrade. Architecture evaluations help software developers to ensure that the architecture will be able to meet the quality requirements (BOSCH, 2000). For this reason, some approaches to evaluate software architectures have been proposed. Bosch (2000) identified four main groups for these evaluation approaches: (i) experience-based methods, which are based on previous experience and domain knowledge of the evaluation team; (ii) simulation-based methods, which typically rely on a high level implementation of the software architecture; (iii) mathematical modeling methods, which use mathematical proofs to evaluate operational quality requirements, such as performance and reliability; and (iv) scenario-based methods, which evaluate a particular quality attribute by creating scenario profiles. Each scenario describes an intended use of the system by means of a concrete description of the quality requirement. These scenarios support the identification of architectural risks and its potential consequences through an efficient and scalable way. Some methods for architectural evaluation are Architecture Tradeoff Analysis Method (ATAM) (KAZMAN; KLEIN; CLEMENTS, 2000), Cost Benefit Analysis Method (CBAM) (KAZMAN; ASUNDI; KLEIN, 2001), and Decision-Centric Architecture Reviews (DCAR) (HEESCH et al., 2014).

2.2.1 Brief History

In 1980’s, software architecture emerged as an important research area for the study of large-scale structures of software systems. As a result, research on this discipline has matured to encompass broad explorations of notations, tools, and analysis techniques, offering concrete guidance for large and complex software design and development (SHAW; CLEMENTS, 2006).

Maturation of software technologies typically takes from fifteen to twenty years to enter in widespread use, in which it evolves through six typical phases (REDWINE JR.; RIDDLE, 1985): (i) basic research; (ii) concept formulation; (iii) development and extension; (iv) internal enhancement and exploration; (v) external enhancement and exploration; and (vi) popularization. Currently, software architecture research is on the last phase, which began in 2000 (SHAW;
CLEMENTS, 2006). But there is still a great research community, as the demand for large and complex systems is still increasing. Both European Conference on Software Architecture (ECSA) and Working IEEE/IFIP Conference on Software Architecture (WICS) are the two actively main events on the area.

### 2.2.2 Software Architecture and System Qualities

The desired quality attributes of a system can not be assured to be on the final product by the software architecture itself. That is, the software architecture does not exactly satisfy non-functional or quality attribute requirements. Instead, the architecture enables the remaining phases of software development to be done concerning the satisfaction of system qualities, which, in turn, allows the final product to exhibit its required quality attributes (BASS; CLEMENTS; KAZMAN, 2012). Some examples apply: if a system requires high performance, then its necessary to manage the communication frequency, time, and volume among its structures. To enable a system to be easily modifiable, then the design must follow high cohesion and low-coupling approaches, so further changes to the system do not reach far consequences. If security is priority, it is required to manage the communication among system structures and which elements are allowed to access which information. These and other similar examples are supremely architectural (BASS; CLEMENTS; KAZMAN, 2012). But, as said before, the architecture "enables" the system to inhibit its quality attributes, that is, the architecture can not guarantee quality, i.e., poor downstream design or implementation can always undermine an adequate architectural design (BASS; CLEMENTS; KAZMAN, 2012).

Additionally, since the architecture is normally designed in early phases of the development process, it can be studied to support the prediction of quality (BASS; CLEMENTS; KAZMAN, 2012). This is useful because enable early design changes, which are less expensive and requires less efforts. In this sense, architectural evaluation methods can be conducted to support top-down insight into the attributes of software product quality that is made possible (and constrained) by software architectures (BASS; CLEMENTS; KAZMAN, 2012).

### 2.2.3 Architectural Process

The architecture is normally designed right after requirements phase, and the general process has three main activities (HOFMEISTER et al., 2005): architectural analysis, synthesis and evaluation. Architectural analysis articulates architecturally significant requirements (ASRs) accordingly to architectural concerns and the context. In architectural synthesis, architectural decisions are proposed to address the requirements of architectural analysis. Finally, architectural evaluations help verifying that architectural decisions made are the right ones to enable the requirements to be satisfied. These activities are not executed sequentially, instead they are conducted repeatedly at many levels of granularity until the architecture is complete and evaluated (HOFMEISTER et al., 2005).
The overall process is shown in Figure 6, concerning the three main activities and their input/output artifacts. Following, relevant information about architectural analysis and evaluation is presented, since they are important for this Master’s Project.

![Figure 6 – Architectural design activities (HOFMEISTER et al., 2005)](image)

**Architectural Analysis**

Architectural analysis has as its main objective the definition of problems that the architecture must solve. For this, (i) architectural concerns and (ii) the context are examined and ASRs are then gathered. Architectural concerns (i) consists of interests in a system relevant to one or more of its stakeholders (ISO/IEC/IEEE 42010., 2010), including system considerations on quality attributes such as performance and reliability. Most of them are normally expressed as system requirements, but can be expressed in form of mandatory design decisions such as the use of existing standards. Requirement documents are also source of architectural concerns, but they are not likely to be of much use since their main function is to provide functional requirements (BASS; CLEMENTS; KAZMAN, 2012). The context (ii) is all circumstances of developmental, operational, political, and other influences upon a system (ISO/IEC/IEEE 42010., 2010). This includes business goals (e.g., reduce production costs, improve market position), characteristics of the organization (e.g., skill of developers, tool available for software development), and the state of technology (HOFMEISTER et al., 2005). The distinction of concerns and the context is sometimes whether it is specifically desired for the system, in this case its a concern, or is instead, a general characteristic or goal of the organization or a stakeholder, that is not desired for the system (HOFMEISTER et al., 2005).

The ASRs gathered/outputted in this activity are requirements that will profoundly affect the architecture, molding it (BASS; CLEMENTS; KAZMAN, 2012).

**Methods**

There are three major approaches to gather ASRs during architectural analysis (BASS; CLEMENTS; KAZMAN, 2012). The first and most obvious one is to look at requirements document. Although this approach is not of much help, some ASRs can in fact be gathered off
these documents. Secondly, ASRs may be gathered by interviewing stakeholders. A method for this approach is Quality Attribute Workshop (QAW), which will be hereafter described. Last but not least, ASRs can be gathered from business goals. In this approach, the focus is on what firstly motivates the development of a software system. Pedigreed Attribute eLicitation Method (PALM) is used in this sense to derive quality attribute requirements (ASRs) from business goals. Following, both QAW and PALM are described.

The QAW is a facilitated method that engages system stakeholder (e.g., end-users, database administrators, acquirers, software engineers, etc) during early development of the system, in order to discover quality attributes that will drive the design phase of software-intensive systems. The number of stakeholders vary from five to thirty for a single workshop. Each of them receive a handbook providing examples of quality attribute taxonomies (i.e., quality models), questions, and scenarios. If necessary, this handbook may be customized to the system’s domain, therefore, its contents should be appropriate for the domain and the architectural detail level (BARBACCI et al., 2003). During the workshop session, every stakeholder is encouraged to intensively participate on the discussion, asking questions if necessary. However, the facilitator may have to interrupt if the discussion when it is taking so much time or when it is not focused on QAW outcomes. The QAW involves eight steps, as follows:

**Step 1: QAW Presentation and Introductions.** This step consists of explaining how QAW step works. This may be done using a standard slide presentation. The participants introduce themselves, providing their background role in the organization, and their relationship to the system being built.

**Step 2: Business/Mission Presentation.** In this step, company representatives present business and/or mission drivers for the system. This is done in about one hour and should additionally address high-level functional requirements, constraints, and quality attribute requirements. Relevant information is then captured by facilitators to uncover quality attribute drivers. Quality attributes that will be refined in further steps are largely derived from the business/mission needs presented during this step.

**Step 3: Architectural Plan Presentation.** This step consists of the presentation of any high-level system descriptions, context drawings, or artifacts that have been created to describe system’s technical details. This includes plans and strategies to satisfy business and mission requirements, key technical requirements and constraints such as operating systems, hardware, and standards, and existing diagrams. During this time, facilitators continue to capture key aspects of the presentation.

**Step 4: Identification of Architectural Drivers.** In this step, the facilitators should excuse the group for a 15-minute break. During this time, every information capture in steps 2 and 3 are compared and consolidated. After this, when the stakeholders are back, the facilitator will share their list of key architectural drivers and ask the stakeholders for
further corrections, aiming at achieving a consensus on a distilled list of architectural drivers. This list will help focus on stakeholders during next step.

**Step 5: Scenario Brainstorming.** With a list of architectural drives at hand, the facilitator initiate a brainstorming process in which stakeholders generate and suggest scenarios, which should be reviewed by the facilitator to assure they are well formed. The scenarios that each stakeholder express represent their concerns with respect to the system, and are expressed in a round-robin fashion. It is advisable that scenario generation is made following any guideline.

**Step 6: Scenario Consolidation.** In this step, similar scenarios are consolidated and merged, as long as the stakeholders that proposed them agree and feel that their scenarios will not be diluted during the process. This consolidation is made from voting. Facilitators should make every attempt to reach a majority consensus with stakeholders before merging scenarios. Actually, only a few scenarios are merged.

**Step 7: Scenario Prioritization.** The prioritization in this step is made from voting. Each stakeholder receive 30% of the total number of scenarios generated after last step. The votes are then allocated in two steps, in a round-robin fashion. During each one, stakeholders allocate half of their votes. The votes are counted, and the scenarios are prioritized accordingly.

**Step 8: Scenario Refinement.** Finally, accordingly to the amount of time left, the top four or five scenarios are refined in more detail. Thereafter, they are further elaborated and documented with the following information: (i) stimulus; (ii) response; (iii) source of stimulus; (iv) environment; (v) artifact stimulated; and (vi) response measure. Stimulus (i) is the condition that affects the system. Response (ii) is the answer for the stimulus. Source of stimulus (iii) is the entity that generated the stimulus. Environment (iv) is the condition under which the stimulus occurred. Artifact stimulated (v) is self-explanatory. Response measure (vi) is the response by which the system will be evaluated. This step is repeated until time runs out or the highest priority scenarios have been refined.

PALM has two purposes: give a clear and full picture of the operative business goals to architects and uncover difficult requirements that may be not necessary since they do not support any important business goal. The output of PALM is a prioritized list of business goals and the associated quality attribute requirements derived from them. This method can be carried in two days, following seven steps:

**Step 1: PALM overview presentation.** Here, an overview of PALM is given, including the problem it solves, its steps, and expected outcomes.

**Step 2: Business drivers presentation.** A representative of project management presents briefly the business drivers, including the most important business goals from both customer and
development organizations. The business drivers presented here are expected to be broader and less precise, and are refined along other elicited goals in next steps.

**Step 3: Architecture drivers presentation.** The architect briefly presents driving (shaping) business and quality attribute requirements.

**Step 4: Business goals elicitation exercise.** Using standard business goals to guide the discussion the most important goals are captured for the system. They are elaborated and expressed as business goal scenarios. Thereafter, each goal derived should have as the goal-object, something tied to the system under development. A goal with other goal-objects must be translated into goals tied to the current system.

**Step 5: Identifying potential quality attributes from business goals.** In this step, for each goal scenario, the participants describe a quality attribute that would help achieve it. If the quality requirement is already a current system requirement, it is tagged as a non-risk. Else, it is tagged as a risk. ASRs can be tentatively identified in this step.

**Step 6: Examination of existing quality attribute drivers.** In this step, for each architecture driver established in Step 3, business goals to support them are identified. If it supports none or support a business goal that is low priority, that is tagged as a risk. Otherwise, it is tagged as a non-risk. In this case, the quality attribute requirement is strengthened by asking its source about the requirement rationale.

**Step 7: Exercise conclusion.** Finally, results are reviewed and participant feedback is gathered.

### 2.2.4 Architectural Evaluation

In this activity, the goal is to ensure that the architectural design decisions made on architectural synthesis are the right ones to enable the achievement of both functional and non-functional requirements. For this, the candidate architectural solutions are verified upon ASRs outputted from architectural analysis. At the end of multiple iterations, which consist of acceptance or disapproval of candidate solutions, the validated architecture is expected as result (HOFMEISTER et al., 2005). Candidate architectural solutions may present alternative solutions, and/or may be partial solutions (i.e., fragments of an architecture) that reflect design decisions about the structure of software. This includes include information about the design rationale, that is, commentary on made decisions, which of them were considered and rejected, and to which requirements they are linked (HOFMEISTER et al., 2005).

**Architectural Evaluation Methods**

Following, two methods for architectural evaluations are described. The first one is Architecture Tradeoff Analysis Method (ATAM), which was proposed by Kazman, Klein and Clements (2000), and Decision-Centric Architecture Reviews (DCAR), recently proposed by Heesch et al. (2014) to overcome high efforts required by previous methods, including ATAM.
2.2. Software Architecture

ATAM aims at assessing the consequences of architecture decisions in light of quality attribute requirements, providing means to detect areas of potential risk within the architecture of systems. Therefore, it is critical to have precise characterizations for each quality attribute (KAZMAN; KLEIN; CLEMENTS, 2000). It can be applied during early design phases or on legacy systems. The results, commensurate to the level of architectural specification provided as input. In addition, ATAM records any risks, sensitivity and trade-offs points during its application. Risks are architecturally important decisions that have not been made (e.g., the decision on which database management system will be used) or decisions that have been made but their consequences are not fully understood (e.g., the architecture team decided to include an operating system portability layer, but is not sure about which functions need to be on this layer). In turn, sensitivity points work as parameters to which some measurable quality attributes response is highly correlated to. One might determine that overall performance in the system is highly correlated to the performance of one specific communication channel, and the availability in the system is also highly correlated to the reliability of that same communication channel. A trade-off point is then found when increasing the speed of this specific communication channel, its reliability is negatively affected. The speed of that channel is, therefore, an trade-off point. The overall concept of ATAM is shown in Figure 7.

![Figure 7 – ATAM Concept interactions (KAZMAN; KLEIN; CLEMENTS, 2000)](image)

The conduction of ATAM evaluation is structured around the idea that architectural styles are the main determiners of architectural quality attributes, which are, in turn, derived from business goals. The steps of ATAM are:

- Presentation

**Step 1: Present the ATAM.** Here, the method is described to the assembled stakeholders (e.g., typically customer representatives, the architect or architecture team, user representatives, maintainers, administrators, managers, testers, integrators, etc.)
**Step 2:** Present business Drivers. The business goals are presented by the project manager. These goals will influence or even drive the development and, hence, what will be the primary architectural drivers.

**Step 3:** Present architecture. The architect present the architecture, and how it is supposed to address the business goals previously presented.

- Investigation and Analysis

**Step 4:** Identify architectural approaches. Architectural approaches are identified but not analyzed, yet.

**Step 5:** Generate quality attributes utility tree. The most important quality attributes are elicited, specified down to the level of scenarios, annotated with stimuli and responses, and prioritized.

**Step 6:** Analyze architectural approaches. Based on the quality attribute tree defined of Step 5, the architectural approaches identified are finally analyzed. In this step, architectural risks, sensitivity points, and tradeoff points are identified.

- Testing

**Step 7:** Brainstorm and prioritize scenarios. Considering the scenarios of Step 5, a larger set of scenarios are elicited from the entire group of stakeholders. Thereafter, these sets are prioritized though a voting process involving the entire stakeholder group.

**Step 8:** Analyze architectural approaches. This step reiterates the Step 6, but the highly prioritized scenarios of Step 7 are used as test cases to analyze the architectural approaches determined so far. This will help uncovering additional architectural approaches, risks, and sensitivity and tradeoff points.

- Reporting

**Step 9:** Present results. Based on the information collected during the process (i.e., styles, scenarios, attribute-specific questions, the utility tree, risks, sensitivity points, trade-offs), the ATAM team presents the findings to the assembled stakeholders and writes a report with this information, which may include mitigation strategies to improve the architecture.

All of those steps are typically carried in two phases. The first phase concentrates on eliciting architectural information and analyzing it. The second one consists of eliciting the point of view of stakeholders and verifying the results of the first phase. Both steps takes up to three full days of work to be conducted. This motivated Heesch et al. (2014) into developing DCAR, a lightweight software architecture evaluation method.
Initially, the main differences between ATAM and DCAR are related to the number of stakeholders taken into the evaluation process, and the time it takes for completion. DCAR requires the participation of the lead architect, one or two members of development team (both of them must have different roles and responsibilities), and somebody to represent both customer and management perspectives. Therefore, in total, DCAR requires the presence of three to five members, and less than three person-days, plus two-person days for review team. An external review team may be used but, in this case, it is highly recommended that they are very experienced in designing software architectures in the same domain as the architecture under evaluation.

As the name suggests, this method is based on architectural decisions, meaning that it can only be applied after some of them have been made. The evaluation process starts when stakeholders select a set of these architectural decisions, based on decision forces related to the project and the company. Decision forces are non-trivial influences on the architecture (e.g., quality requirements, constraints, risks, political or organizational-considerations, personal preferences, experience, and business goals). That said, DCAR is conducted in nine steps, as Figure 8 shows. Following, a brief description of each step is provided.

**Step 1: Preparation.** In this step, a date for the evaluation session is defined and the stakeholders are invited. Additionally, the chief architect prepares a presentation of the architecture, concerning architectural requirements, high-level views of the architecture, and approaches and technologies used on the architecture. In turn, the management and customer representative prepares a presentation describing the software product and its domain, the business environment, and other architectural drivers. The presentations are sent to the review team prior to the evaluation session so they may record a list of architectural decisions and decision forces for the session.

**Step 2: DCAR Introduction.** The steps of DCAR are presented along with the day’s schedule. Here, information such as evaluation’s scope and participants’ roles and responsibilities are described.

**Step 3: Management presentation.** Here, the focus is on the identification of business-related decision forces that must be considered for the evaluation. The review team notes any potential forces during the presentation and ask questions to elicit additional ones.

**Step 4: Architecture presentation.** The goal of this step is to give all the participants a good overview of the architecture. This step is supposed to be highly interactive, as it is necessary for the review team to complete their list of decisions and forces done in Step 1.

**Step 5: Forces and decision completion.** In this step, architecture decisions and their relations are clarified, and the list of relevant forces of these architecture decisions is complete. To support it, any UML tool may be used to model relations between these elements.
Step 6: Decision prioritization. Normally, the number of decisions elicited in previous steps is too large. Hence, it is necessary to prioritize which of them are going to be reviewed. For this, 100 points are given to the participants, that must distributed them between the decisions. Done it, the points are summed up and the top decisions are chosen. It is recommended seven to ten decisions for a half-day session.

Step 7: Decision documentation. The chosen decisions are then documented and each participant selects two or three of them, based on the knowledge they have about them. The documentation describes applied architectural solutions, the problem or issue it solves, known alternative solutions, and the forces that must be considered to evaluate the decision.
2.3. Decision evaluation. In this step, the selected decisions are presented, accordingly to their priority, by the selected participants to which them were assigned. Then, additional forces are identified and documented in challenge against the chosen solution. All participants discuss the forces, evaluating the pros and cons of the solutions and finally votes if the solution is good, acceptable, or has to be reconsidered. Rationale for the votes are also given.

Step 9: Retrospective and reporting. Finally, every the artifacts generated throughout the session are collected as input for the final report, which is further refined and verified for refinement.

2.3 Software Quality

Software quality has been on top of the objectives of software engineering since its beginnings. The increasing use of software in many areas of the society was and is still responsible for a huge transformation on people’s day-to-day. This transformation is very important in many aspects; however, brought some problems that emerged from different software issues, resulting in a final product that does not satisfy the customers, and was, in critical domains, the cause of catastrophic events. Software quality is a complex and multifaceted concept (Garvin, 1984) related to meeting the system requirements, the degree to which a system satisfy its customer and users, the totality of characteristics of a system to meet stated and implied needs, etc. There is no unique and wide-accepted definition for software quality, but it is expected that the final product presents a high level of quality. This leads the software engineering community to research this area and present diverse approaches to support engineers in the development of high-quality software.

In this context, software QMs are one of the many approaches discussed. They provide a better understanding of QAs and sometimes operationalize them in a practical and useful way, which enable direct measurement of QAs. Following, concerns about QAs are presented.

2.3.1 Quality Attributes

Some of the most important requirements concern the system’s QAs. These requirements are characteristics of the system, as opposed to functionality that describes what the system does. Example QAs include usability, maintainability, performance, and reliability. Sommerville (2010) notes that functional requirements are services the system should provide, while non-functional requirements (or QA requirements, understood as requirements for QAs) are constraints on the functions offered by the system. Different communities address QAs with different nomenclatures but we choose “quality attribute” as this work is contained within the software architecture area and that is how the community addresses the many *bilities. Other communities address the
same concept as quality factors, quality characteristics, software attributes, non-functional requirements, quality requirements, etc.

QAs by themselves do not have any significant meaning unless there is prior knowledge by who is looking at them. Of course the meaning of many QAs can be inferred because they are employed in common sense. Security, for example, has the same meaning in the context of a software system and a house (protect the subject from unauthorized access), although the many approaches for ensuring security in both contexts are completely different. However, particularities in each context will require at least a minimal clarification about the QAs been addressed, which means that it is important to provide unambiguous definition for QAs. This is done by the authors of many QMs discussed in Section 2.3.3 (McCall, Richards and Walters (1977), Boehm, Brown and Lipow (1976), ISO/IEC 9126 (2001), ISO/IEC 25010 (2011)).

In order to address QAs in a more concrete way, Bass, Clements and Kazman (2012) and Bachmann et al. (2005) use QA scenarios to describe QA requirements. This approach is similar to use cases employed for functional requirements specifications and can be used to support the understanding of quality goals from the perspective of different kinds of stakeholders. Also, scenarios helps operationalize QAs requirements, making them measurable in an objectively or subjectively manner. A quality attribute scenario is composed by the specification of the following elements:

| **Stimulus**: is an event that arrives at the system and requires it to answer adequately. It can be, for example, a security attack, a change request issued to modify system features, a communication between two servers to exchange information, etc. |
| **Source of stimulus**: is the entity that generates the stimulus and can be, for example, system stakeholders in general or other software systems. |
| **Environment**: a stimulus occurs under some conditions, which can be related to the state of the system (e.g., normal operation, overloaded) or system life-cycle phase such as design, testing or development. |
| **Artifact**: is the subject of the stimulus within the scenario. It can be the system itself, the source code, the data hold within the system, intermediate systems, etc. |
| **Response**: is the reaction of the artifact to the stimulus. For example, it can be an access denial for an unauthorized entity, data resulted from some processing, a change implemented in the source code, etc. |
| **Response measure**: is the way in which the answer given for stimulus of the scenario is measured. For example, the time the system takes to process a bunch of data, unavailable time after a failure, time to make a change in parts of the system, etc. Normally it is specified as the expected measure for stimulus instead of the metric itself, and can be an objective or
2.3. Software Quality

A very important concern about QAs is that they can present relations or interactions among them that is of utmost importance during software development, specially for software architecture design. The most common relation of concern among attributes is named trade-off. A quality attribute trade-off is a relation between two quality attributes that are incompatible and can not be individually fully optimized for a given system, which means that designers have to consider a balance between both to satisfy stated requirements (BARBACCI et al., 1995). Knowing trade-off relations between quality attributes is useful to support design decisions in general. The Figure 9 shows how a trade-off is supposed to happen by the interaction of three QAs (performance, dependability, and security). Choosing to maximize one QA (local optimum) implies that the others can not reach their local optimum. In this situation, software engineers have to find a balance point (global optimum) that satisfies the quality requirements for a given system.

![Figure 9 – Graphical view on trade-offs between performance, security and dependability (BARBACCI et al., 1995)](image)

Considering its experience with CES, Bate (2008) proposed a process based on the Goal Structuring Notation (GSN) to understand and evaluate design trade-offs. The process is...
composed by nine activities distributed in four main stages, as described below:

Stage 1: Presenting the current design.

Part (a) - Produce Initial Architecture Model: in this part, it is necessary to produce the architectural design using a model decomposed to an uniform level of abstraction. Any approach that allow one to identify components and their relations can be employed for this goal, as is the case of simple blocks and lines, and UML diagrams.

Part (b) - Improve Design: any changes applied to the model produced in Part (a) implies in performing Stages 1 to 3 of the process again.

Stage 2: Producing an argument for the key objectives: key objectives such as ease of integration and verification, and key properties of the system such as life cycle cost and quality attributes are decomposed into detailed design requirements. Further, these requirements are structured along with the appropriate context and identifying available design choices, forming an argument with GSN.

Stage 3: Extracting information from the argument.

Part (a) - Extract Design Choices: it is possible to further decompose the objectives from the argument into design choices until a solution is reached. The decomposition of some design choices might be impossible due to design restrictions.

Part (b) - Extract Assessment Criteria: with a suitable design argument in hands, evaluation criteria (design objectives) need to be extracted for the next stage.

Part (c) - Evaluate Architectural Design: in this part, it is necessary to determine how each evaluation criteria will be converted into a value that will support comparison among design decisions. To do this, it is necessary to understand the degree to which different solutions meet the objective, deriving a metric. Its value must be normalized and preferable quantitatively evaluated rather than by subjective means.

Part (d) - Scenario-Based Assessment: this assessment part is one of the two approaches presented by the authors to assess the evaluation criteria established in the last part. Scenario-based assessment takes place when Stage 3 (e) can not be easily applied.

Part (e) - Baseline Assessment: without any scenario being applied, baseline assessment considers existing techniques to evaluate properties and objectives. For example, schedulability analysis can be employed to evaluate timing constraints.

Stage 4: Decomposing the design.

With a suitable design in hands, it is necessary to decide whether and how it should be decomposed to a next level and recommence at Part 1.
Part (a) - Derive Interface Requirements: in this part, interface requirements (i.e., constraints that exist between different parts of the design) can be represented by assumptions and context within design arguments.

Part (b) - Produce Refined Architectural Model: here, design choices are considered to produce the refined architectural model.

Other approaches for addressing QA trade-offs within software community can be found in the systematic mapping conducted by (BARNEY et al., 2012).

(FEITOSA et al., 2015) empirically investigated trade-offs among QAs in the context of open source CES and compared them with trade-offs in non-critical open source projects. They considered the implemented software architecture (i.e., source code) of twenty-one projects developed in Java. The authors systematically monitored the evolution of source code of each project based on the official releases published in online version control systems. For each release, they computed a set of metrics for critical QAs (correctness, security and performance) and non-critical QAs (effectiveness, extendibility, flexibility, functionality, reusability and understandability). Analyzing the deltas for each project and each quality attribute, the authors identified QA trade-offs in the following way: if the computed value for a given QA decreases from version 1 to version 2 but increases for another QA, it is identified as a trade-off. They did it for each version within each project and merged the results, giving the following findings:

- Trade-offs among QAs in CES
  - reusability negatively affects extendibility; and
  - performance negatively affects reusability.

- Trade-offs among QAs in non-critical projects
  - correctness negatively affects all QAs; and
  - security is affected by all critical QAs (correctness and performance).

- Comparison between the two groups
  - non-critical QAs do not affect critical QA; and
  - security negatively affects correctness; and
  - correctness negatively affects performance.

A particular challenge with QAs is that system-wide approaches are required to satisfy them. These approaches are defined at the system architecture level and not at the component level. Clements, Kazman and Klein (2002) state the following: "Modifiability, performance,
security, availability, reliability – all of these are precast once the architecture is laid down. No amount of tuning or clever implementation tricks will wring any of these qualities out of a poorly architected system.” However, they cannot be fully tested until the software has been built. This creates a gap between satisfying QAs during design, and when validating QAs during testing.

The satisfaction of critical QAs is a first-class concern. Measures to accommodate critical QAs must be taken early, because critical QAs are tightly linked to the architecture of the system (BASS; CLEMENTS; KAZMAN, 2012). A system’s existing architecture configuration may be highly compatible with certain measures for critical qualities (e.g., patterns for safety or reliability), or may be of no help whatsoever. Thus, the architecture influences how we can implement measures to achieve critical QAs. This means measures taken to improve critical QAs can be implemented with little or no change to the architecture, or may require significant changes. Such changes, besides requiring significant effort to implement, can dilute the original intent of the architecture.

The challenge of designing a critical system architecture is tremendous. Ideally, the architecture supports the measures for critical QAs, and they are implemented within the architecture. However, other functional and non-functional requirements also shape the architecture, resulting in an architecture that does not perfectly support critical QAs. This forces the architect to make trade-offs, to select measures for critical QAs that are most compatible with the architecture, or to modify the architecture to accommodate critical QAs. In order to architects make these decisions correctly and efficiently, they need information about how critical QAs and the architecture interact.

2.3.2 Quality Metrics

Measurement lies at the heart of many systems present in the society (FENTON; PFLEEGER, 1998). In the economic area, measurements determine the price of things and the increase in payment. For avionics, measurement allows radar systems to detect airplanes even if the pilot’s vision is obscured for some reason, and therefore helps pilots to avoid accidents. In medical systems, measurement is used by doctors to diagnose precisely ill patients. Many other examples could be given. In software engineering, the area of software metrics has been under research since its early days (WAGNER et al., 2012). The usage of metrics support a better and faster development of systems (WAGNER et al., 2012).

In the context of software quality, a quality metric (or software quality metric) is a function whose inputs are software data (e.g., source code, logs, artifacts) and whose output is a single numerical value that can be interpreted as the degree to which a system possesses a given QA (IEEE., 2009). These metrics can be used throughout software life cycle to evaluate the quality of a given software artifact, reducing the subjectivity of software quality and being a quantitative basis for early and late decisions. More specifically, organizations can use quality metrics to (IEEE., 2009): (i) Assess achievement of quality goals; (ii) establish acceptance criteria
and standards; (iii) evaluate the level of quality achieved against the established requirements; (iv) detect anomalies or point to potential problems in the system; (v) predict the level of quality that will be achieved in the future; (vi) monitor changes in quality when software is modified; and (vii) assess the ease of change to the system during product evolution.

As an example of quality metric concerning usability, we may look at the approach proposed by ISO 9126-2 (ISO/IEC 9126-2, 2001). In order to measure the usability of a system, this attribute has been decomposed into three lower-level attributes: (i) learnability, (ii) understandability, and (iii) operability. Measuring these attributes, we are fully measuring, in theory, the degree in which usability is present on a system. Therefore, to determine this degree, the standard proposes some metrics to measure each of these attributes. In turn, to measure learnability (i), six metrics are proposed. One of them is ease of function learning. This metric aims at determining the time a user takes to learn to use a system function. For this, a test is conducted and the behavior of the user is observed. The metric itself is the mean time the user takes to learn a function correctly. The less time the metric outputs, the better. This metric is linked to learnability (i) which is in turn linked to usability. These attributes comes from a QM proposed by ISO 9126-1 (ISO/IEC 9126, 2001).

The selection or definition of rigorous quality metrics is a requirement to obtain useful QMs for doing reliable comparisons (FRANCH; CARVALLO, 2003) among different versions of the same software product or different software products. A rigorous and useful quality metric is the one that have two very important properties (WAGNER et al., 2012): reliability and validity. In this context a reliable metric means that the measure gives almost the same result every time it is measured. In turn, validity means that its value corresponds correctly to the attribute it is measuring. The possibilities for the reliability and validity of a measure is shown in Figure 10. A measure is neither reliable nor valid if it produces different values every time and the values do not corresponds to the attributes of the object being measured. To be reliable, a measure must produce similar results in every measurement but the values do not corresponds to the reality. The ideal scenario is when both reliability and validity are properties of the metric. However, defining metrics is not an easy task, since validity is often difficult to show and reliability can be problematic for manually collected metrics which are often subjective (WAGNER et al., 2012).

More about this relationship between QMs, attributes and metrics are given in Section 2.3.3.

### 2.3.3 Quality Models

According to the Standard ISO/IEC 25000. (2005), a QM is a defined set of characteristics (or QAs), and of relationships between them, which provides a framework for specifying requirements and evaluating quality. This framework supports the specification and evaluation of software and software-intensive systems from different perspectives by those associated with their acquisition, requirements, development, use, evaluation, support, maintenance, quality assurance
Chapter 2. Background

Figure 10 – Reliability and validity of measures (Wagner et al., 2012)

QMs have been subject of research by the software engineering community since its early days, constituting the foundations of software product quality (Wagner, 2013), and a broad topic of research. Deissenboeck et al. (2009) classify QMs accordingly to their purpose: a definition QM (i) focuses on what is quality and/or how it could be organized in a hierarchical way that emerges from a divide and conquer approach, considering non-overlapping attributes; assessment QM (ii) are used to evaluate the software product or software artifacts generated throughout the software life-cycle; and prediction QM (iii), which are considered more advanced since they can be used to process data and therefore predict the quality of a software product before its implementation phase. This classification is shown in Figure 11.

Many QM were proposed in the software engineering literature to support the evaluation of software products (Singh; Kannojia, 2013): McCall Quality Model (1977) (MC-
2.3. Software Quality

CALL; RICHARDS; WALTERS, 1977), Boehm’s Quality Model (1976) (BOEHM; BROWN; LIPOW, 1976), COQUAMO (1987) (KITCHENHAM, 1987; KITCHENHAM; PICKARD, 1987), FURPS (1987) (GRADY; CASWELL, 1987) Dromey’s Quality Model (1995) (DROMEY, 1995), ISO/IEC 9126-1 (2001) (ISO/IEC 9126, 2001), QMOOD (2002) (BANSIYA; DAVIS, 2002) and the recent ones ISO/IEC 25010 (2011) (ISO/IEC 25010, 2011) and QUAMOCO base model (2012) (WAGNER et al., 2012).

Essentially, these models present different perspectives in which quality should be considered. McCall’s model has three major perspectives to identify and define quality: Product operations, revision and transition. Each one of them have their own QAs. For example, product revision is composed of maintainability, flexibility and testability. Boehm, Brown and Lipow (1976) defined three levels of characteristics of quality code. The first and most general level is composed of general utility, as-is utility and maintainability. These are further decomposed into intermediate level characteristics such as portability, reliability, efficiency, understandability, and more. Finally, these last characteristics are decomposed into at least one non-exclusive lower level (or primitive) characteristic, such as accuracy, legibility, accessibility, and robustness/integrity. All of these characteristics are further discussed by Boehm, Brown and Lipow (1976) under the measurement of the quality of the code, where some metrics are presented and evaluated accordingly to their correlation to quality, potential benefits to developers, quantifiability, ease of developing an automated measurement tool and its completeness.

Standardization initiatives bring the general software QMs ISO 9126, in 2001 and ISO 25010, in 2011 to the software industry. These models are part of a quality evaluation framework that provides a taxonomy of QAs, their definition in the context of software, and metrics to measure the QAs of software systems.
2.3.4 **Quality Model Establishment**

In general, QMs can be established through two main approaches (CALERO et al., 2007): (i) from scratch (see: (BOEHM; BROWN; LIPOW, 1976; MCCALL; RICHARDS; WALTERS, 1977)) and (ii) tailored upon existing QMs (see: (DROMEY, 1995; KHARCHENKO; GALAY; YATCYSYN, 2011; ESCOBAR; LOSAVIO; ORTEGA, 2013)). The first approach requires more efforts and does not provide any kind of reuse. QMs built following this approach normally represent the first attempts to model quality and are classical references nowadays. However, recent QMs are built focusing on knowledge reuse, through the use of meta-models which based the establishment of other QMs or direct selection of QAs, based on the most important requirements of a given domain. Nevertheless, the building of QMs is still a complex activity, endangered by many factors, such as poor description of the domain, lack of ability to identify quality entities, and so on (FRANCH; CARVALLO, 2002). However, when these QMs are made, they become a powerful tool which provides a framework to support the evaluation of software systems (FRANCH; CARVALLO, 2002).

Different guidance to develop those QMs have been proposed aiming at different purposes. Franch and Carvallo (2002) defined a methodology for building ISO/IEC-based QMs, comprising the establishment of QAs, their hierarchy and relationships, and metrics to these attributes.

Andreou and Tziakouris (2007) tailored the ISO 9126 standard (ISO/IEC 9126, 2001) to build a QM to assess and evaluate original software components, which are related to Component-Based Software Engineering (CBSE). In addition, they provided guidelines to customize the ISO for software components. Additionally, some experiences on establishing and adapting software QMs can be found in Calero et al. (2007), Franch et al. (2008), Behkamal, Kahani and Akbari (2009), Villalba, Fernández-Sanz and Martínez (2010), Kläs, Lampasona and Münch (2011).

2.3.5 **Domain-Oriented Quality Models Development (DUMOD)**

Villalba, Fernández-Sanz and Martínez (2010) conducted a systematic literature review related to software quality models aiming at finding a valid method to generate domain-oriented quality models proposed in the literature. The authors included a set of 24 primary studies for analysis and draw some conclusions regarding the establishment of quality models.

The first conclusion is that the first step to construct a valid model is to use standards and other literature as source of relevant quality attributes and related concerns. This procedure was performed by the majority of authors of the primary studies found.

The second conclusion address the number of factors (quality attributes) present in the quality model. In order to build a practical and efficient quality model, it is necessary to reduce the number of factors based on the opinion of experts regarding different viewpoints. A broad range of perceptions makes the model stronger and more applicable.

The third conclusion addresses the relevance of quality attributes in the model. It is
important to establish the weights of attributes to ease the prioritization of activities that are more important. For example, the ATAM previously explained contains a step to prioritize the scenarios proposed by stakeholders. Without this step, the method would take much more time (in this case we are talking about days of some stakeholders) to be done, at an increased cost and less effectiveness, since low priority scenarios impose less risks to the architecture being evaluated. The weights can be defined by taking expert’s opinion into account, either by applying the Delphi process or collecting data with questionnaires and then calculating mean values for each attribute.

The fourth conclusion refers to gaps found by the authors in a study that presents a step by step process to develop domain-oriented quality models from scratch. This study is the one of Franch and Carvallo (2003) we cited before. They highlight that this process does not provide two important aspects for developing quality models: (1) there is no method presented to reduce the number of quality attributes, to determine their weights in the model nor to establish trade-off relations among them; and (2) the experts’ opinion and perspectives of other authors are not taken into account during the construction process of the model.

Overall, the authors highlight that existing domain-oriented software quality models are of high value to the knowledge of specific types of applications, as is the opinion of experts. The proposed statistical analysis of collected data from experts are, although, weak in some aspects. Based on their experience, the authors of DUMOD says that it is very difficult to identify the most important quality attributes and their weights using the average and median values from data collected from experts. Additionally, these statistics do not support the reduction of quality attributes, finding out their weights, and the relations among them.

Based on the results of the systematic literature review, the authors identified that a deeper work was necessary to define a methodology to overcome the problems previously stated. In this sense, they proposed the approach called DUMOD (Domain-oriented qUality MOdel Development) aiming at facilitating the definition of relevant quality attributes from standards and literature, experts’ and researchers’ experience or knowledge, and providing a way to validate if the quality attributes identified are relevant for the domain they are being established for. The criteria should be checked to see if they are adequate and reduced to a small number that contains the most relevant ones. Data collected from experts and researchers can be used to help identifying weights for each quality attribute and their inter-relations. The authors also analysed models not based in ISO/IEC 9126 in order to find out a method for analyse collected data, and found an approach used for evaluating the quality of services regarding satisfaction in sociological and marketing studies. In this approach, data is collected by experts using questionnaires and multivariate statistical analysis is applied to reduce the number of attributes. This approach was adapted to be used in the development of quality models.

In the following, we explain the overall process of DUMOD that contains three main phases, which consists of a set of activities. We describe each activity separated by phases in the
Phase 1: Initial Analysis

a) Definition and analysis of the domain: in this step, the application domain is defined accordingly to the perspective of authors establishing the model. For example, Villalba, Fernández-Sanz and Martínez (2010) defined their application domain in a single sentence: “Network security COTS domain includes any product with the main function of protecting the networks against malicious intrusions”. This definition is used to conceptually guide the following steps.

b) Adaptation of ISO/IEC 9126 standard to specific domain: in this step, the standard ISO/IEC 9126 is analyzed and modified to fulfill some requirements regarding the application domain previously established. Modifications includes the addition, removal or substitution of quality attributes present in the model. Each modification is followed by a rationale presented by the authors of the model, based on their knowledge and other authors.

Phase 2: Definition of Preliminary Model

a) Creation of the initial set of attributes: in order to complement the adapted standard with additional quality attributes, it is desired to review domain related literature, such as international standards, and scientific and technical literature (e.g., research publications, books, and technical reports). Additionally, lessons learned from the evaluation of products in the domain can be useful to bring other quality attributes to the model.

b) Internal review of the initial set of attributes: the initial set of attributes is usually too large, making it difficult to apply the model efficiently. In this step, the set of attributes is reviewed by an internal group of experts to identify and combine duplicated factors, controlling and solving each conflict, and rewriting the ambiguous ones. After this review, the model is ready to be validated and reduced.

Phase 3: Model Validation

a) Design of factors questionnaires: in this step, a questionnaire is designed to collect data from domain professionals regarding preliminary model attributes. Its structure comprises two parts. In the first one, the goal of the study is explained and anonymous demographic data (such as sector of activity, technical profile, experience or professional qualification and experience with products of the application domain) is collected. Demographic information of each respondent can be later used to accept or reject their submitted data. The second part is composed of multiple-item questions for each attribute in the preliminary model. Each question is, actually, a five-point
Likert-type scale that ranges from 1 (strongly agree) to 5 (strongly disagree). In this sense, the respondent shows its agreement regarding the inclusion of each quality attribute in the preliminary model. A question that asks the utility of the quality model and a free space to allow respondents to include additional quality attribute to the model finishes the questionnaire.

b) Data collection: in this step the questionnaire developed is applied to the audience it was developed for. This can be done using online survey services such as SurveyMonkey\(^1\), Google Forms\(^2\) and LimeService\(^3\). Data from respondents who do not fulfill the expected profile are excluded in this step.

c) Data analysis

i. Preliminary analysis: this step consists of doing descriptive statistics (means, standard deviation, median, minimum and maximum, and absolute and relative frequencies) of all variables collected with questionnaires. It also includes calculation of correlation matrix, which offers information about factorial analysis applicability. Correlations higher than 0.3, significance levels and determinants close to zero show that there are correlated variables. Box plots can also be plotted to help finding input errors, and coefficient of variation can be used to check homogeneity of data. The result of this step is the input for further steps.

ii. Reliability analysis: In this step, it is assessed the degree to which the observed variables measures the real value and is free of error. It is done by grouping sub-characteristics into their parent characteristic and then reducing the number of sub-characteristics for each characteristic by examining corrected item-to-total correlations, and deleting those items which improve reliability alpha coefficient.

iii. Predictive validation: this step consists of running exploratory factorial analysis (EFA) in order to identify the underlying structure of the relations obtaining a predictive validation of the model. This will further support model reduction. The EFA is performed using the method of principal components. Varimax (with Kaiser normalization) and the breaks-in-eigenvalues criterion is used to set the number of factors that will be kept. Factor loadings greater than 0.5 are considered strong, while those with 0.32 or less are removed. To validate the factorial analysis, the Kaiser-Meyer-Oklin (KMO) and Barlett’s test of sphericity should be checked.

iv. Confirmative validation: finally, confirmatory factor analysis (CFA) is performed through structural equation model and maximum likelihood (ML) estimation is used to assess the validity of the model. Then, the model relating the variables and their weights is presented as a diagram.

\(^1\) <https://www.surveymonkey.com/>  
\(^2\) <https://www.google.com/forms/about/>  
\(^3\) <https://www.limeservice.com/>
2.4 Final Remarks

In summary, we can observe that CES are a real, current need and have become quite important to many areas of the society. They are already a leverage for innovation and present almost everywhere, helping molding the contemporary society. Their development must be guided by efficient and dependable methods of production, based on well-established scientific foundations (GILL, 2005). On the other hand, software architecture is a fundamental discipline for system development. Its importance grows as larger and complex systems are. The methods for architectural analysis and evaluation previously described are very important during architectural design, and deal directly with quality concerns, such as quality attribute, therefore, emphasizing the importance of having a good understand of software quality. In this sense, both software architecture and QMs have been more and more investigated, mainly considering that quality needs to be incorporated in software systems, specially those that are safety-critical. In this scenario, incorporating quality in CES is undoubtedly necessary. However, there is a lack of works that propose QM for CES aiming at improving their quality by supporting architectural activities. In the next chapter, we discuss related work in the context of a SM conducted to find QA, quality metrics and QM employed for CES development.
In this chapter, we present three works related to the establishment of quality models for embedded systems. After that, we present the results from conducting and reporting a systematic mapping study performed in the context of CES.

3.1 Related Work

Ahrens et al. (2011) established a specialized QM to conduct the evaluation of software architectures in the context of driver assistant systems. In this study, the authors aimed the evaluation of software architecture of automotive embedded systems, considering that the software architecture is a key factor that influences and is responsible for fulfilling quality requirements of software systems, including scalability, extensibility, and portability as major criteria. The authors, then, argue that until the publication of their results, they did not find approaches for evaluation and measuring of software architecture, therefore, their approach aims at closing this gap. To achieve their objective, a QM was developed with several QAs highly adapted and specialized to the specific needs of the automotive embedded software domain (see Figure 12). Thereafter, eight architectural metrics were defined and presented. Then, a tool was developed to implement every metric, allowing the user to select which metric shall be applied and how they would be weighted to evaluate specific needs or quality requirements for such systems. Finally, to evaluate their approach, a complex and real practical example from a different domain had it software architecture fully refactored using the established metrics in their QM.

Carvalho and Meira (2009) proposed a framework, which includes a QM to support the verification of embedded software components. The framework is also composed of a metrics framework, a certification techniques framework, and a certification process. The QM role in
Figure 12 – The quality model for classifying and evaluating automotive embedded software architectures (AHRENS et al., 2011)

this framework is to determine which quality characteristics should be considered and which sub-characteristics are necessary for component-based development. The model proposed is organized as shown in Figure 13.

Jeong and Kim (2011) established a QM of lightweight components for components in embedded systems. The model aimed to describe correlations between QAs. The selected QAs were divided in five criteria: (i) system quality; (ii) information quality; (iii) function quality; (iv) efficiency quality; (v) and maintenance quality. These criteria were further decomposed on QAs, as following:

- Systems quality: Adaptability, availability, reliability, response time, maturity, fault tolerance, recoverability, and stability;

- Information quality: Completeness, ease of understanding, characterization, relevance, security, up-to-date;

- Function quality: Assurance, empathy, responsiveness, usability, operability, interoperability, and suitability;
3.1. Related Work

- Efficiency quality: Error rate and Operation rate; and

- Maintenance quality: Analyzerability, changeability, MTTR(Mean Time Between Repair), and MTBF(Mean Time Between Failures).

The QM is then established, as shown in Figure 14.

![Figure 13 – Embedded software component Quality Model (EQM) (CARVALHO; MEIRA, 2009)](image1)

![Figure 14 – Quality model for embedded system (JEONG; KIM, 2011)](image2)
3.2 Systematic Literature Mapping

In this section, we report the procedure accomplished in the conduction phase of our systematic mapping, that aims to find which quality attributes, models and metrics have been considered for CES development. The protocol we followed can be found in Appendix A.

3.2.1 Conduction

After adapting and executing the search strings for each search engine, we found 554 studies that match our search terms. After that, we removed duplicated entries, remaining 360 unique studies for first selection. In the first selection, the 360 studies were distributed between three reviewers in a way that each study was reviewed exactly by two reviewers. The 360 studies were assigned to one reviewer (master student), and the two others reviewers analyzed 180 studies each one. Every reviewer applied one or more selection criteria defined in the protocol, after reading the paper’s title, abstract, and if necessary, introduction and conclusion section. After finishing this procedure, the authors gathered to solve divergences in the process of applying selection criteria. If the reviewers applied different criteria for the same study, each reviewer presented its reasons to convince the others and to try to solve differences. If divergence persisted, the third reviewer presented its opinion regarding what criteria should be applied, deciding, therefore, if the study should be included or excluded. At the end of this first selection, 84 studies were included for second selection.

In the second selection, the 84 studies were fully read. We followed the same procedure of first selection regarding study distribution and resolution of divergences. At the end of this step, we included 43 studies for the quality assessment step.

For quality assessment, we divided the group of 43 studies in three. Then, each reviewer assessed every paper it was assigned to answering seven questions related to the general quality of the paper following quality criteria established in the protocol presented in Appendix A. Each question had three possible answers, “no”, “to some extent”, and “yes”, representing each a score of 0 (zero, worse), 0.5, and 1 (one, better), respectively. The sum of scores for each question resulted in a overall score that could range from 0 (zero) to 7 (seven). The greater the score, the better assessed is the paper. After this step, we excluded 20 studies that scored 3.0 or less. The final set of 23 studies included for data extraction is shown in Table 2.

3.2.2 Reporting the mapping results

Data was extracted from the 23 studies following the strategy presented in Section 2.6 of Appendix A. Based on data from extraction phase, we answered the research questions of the systematic mapping. The remainder of this section resolves the research questions.
3.2. Systematic Literature Mapping

![Diagram of the Systematic Literature Mapping process]

**Figure 15** – Process followed to conduct the Systematic Mapping

| ID | Title                                                                 | Reference                        |
|----|----------------------------------------------------------------------|----------------------------------|
| S01| A case study in performance evaluation of real-time tele-operation software architectures using UML-MAST | (ORTIZ et al., 2003)             |
| S02| A complex applications framework supporting tolerant dynamic vehicle dispatching | (BENYAHIA; POTVIN; XU, 2006)     |
| S03| A component model and layered system architecture for reconfigurable CNC systems | (DU et al., 2009)               |
| S04| A design methodology for real-time distributed software architecture based on the behavioral properties and its application to advanced automotive software | (AOYAMA; TANABE, 2011)         |
| S05| An architecture-driven software mobility framework                  | (MALEK et al., 2010)             |
| S06| Analyzing testability on data flow designs                          | (TRAON; OUABDESSE-LAM; ROBACH, 2000) |
| S07| Architectural support for model-driven performance prediction of distributed real-time embedded systems of systems | (CHIPRIANOV et al., 2014)       |
| S08| Architecture description languages for high-integrity real-time systems | (RADJENOVIC; PAIGE, 2006)       |
| S09| Architecture for embedded open software ecosystems                  | (EKLUND; BOSCH, 2014)           |
| S10| Architecture potential analysis: A closer look inside architecture evaluation | (FLORENTZ; HUHN, 2007)          |

**Table 2** – List of all studies selected for extraction activity
Chapter 3. Quality in the context of Critical Embedded Systems

| S11 | Architecture-driven reliability and energy optimization for complex embedded systems | (MEEDENIYA et al., 2010) |
| S12 | Component Based Design of Fault Tolerant Devices in Cyber Physical System | (XU et al., 2012) |
| S13 | Design-driven development of dependable applications: A case study in avionics | (ENARD et al., 2013) |
| S14 | Efficient sensitivity analysis of reliability in embedded software | (MEEDENIYA; ALETI; MOSER, 2015) |
| S15 | Engineering dependability to embedded systems software via tactics | (AL-DAAJEH; AL-QUTAISH; AL-QIREM, 2011) |
| S16 | Evaluating software quality attributes of communication components in an automated guided vehicle system | (MARTENSSON; GRAHN; MATTSSON, 2005) |
| S17 | Evolutionary architecting of embedded automotive product lines: An industrial case study | (AXELSSON, 2009) |
| S18 | From problem to solution with quality attributes and design aspects | (WIJNSTRA, 2003) |
| S19 | Measuring the impact of changes to the complexity and coupling properties of automotive software systems | (DURISIC et al., 2013) |
| S20 | Model-based design analysis of an Avionics Fuel distributed control system | (INSAURRALDE et al., 2008) |
| S21 | Model-Driven Requirements Engineering for Embedded Systems Development | (LONIEWSKI et al., 2013) |
| S22 | Objective evaluation of software architectures in driver assistance systems: Methods-quality model-metrics | (AHRENS et al., 2013) |
| S23 | Quality-driven optimization of system architecture: Industrial case study on an automotive sub-system | (ETEMAADI et al., 2013) |

- **RQ1:** What are the QM established for CES domain?

  From all studies found in this SM, five were classified as quality models [S04, S10, S17, S18, and S22]. In this SM, we only classified primary studies as a quality models when they presented a set of quality attributes in a structured way, as so evidence to measure their importance for the development of CES in specific domains. In other words, there had to be an explicit rationale behind the hierarchy of quality attributes of the quality model. Four of these five studies were conducted in the context of automotive domain [S04, S10, S17, and S22] and one for healthcare domain [S18].

- **RQ1.1:** How have these QM been defined (e.g., based on standards as ISO/IEC 25010, based on accumulated experience, or based on metamodels)?
Two studies [S17 and S18] do not present any approach used to establish their quality models. In both cases, we believe that the authors used their own experience to bring the most important quality attributes for their application domains, and linked these attributes to support the goal of the quality model within their studies. The other three studies [S04, S10, and S22] established their quality model using different approaches: S04 used a metamodel to hierarchically organize quality attributes found in a specification document of a specific system (vehicle dynamic management system); S10 used the QADAG approach that consists of a meta-model to build Quality Attribute Directed Acyclic Graphs. Such a graph was further used to specify the structure of the quality model, linking quality attributes to scenarios and constraints; Finally, S22 tailored their quality model based on previously well-known established quality models in software engineering literature such as (ISO/IEC 9126, 2001; BOEHM; BROWN; LIPOW, 1976; MCCALL; RICHARDS; WALTERS, 1977) and guidelines (LOSAVIO; CHIRINOS; PÉREZ, 2001). The source of each quality attribute within the model was not clearly discussed.

- **RQ1.2: Which purpose the QM has (i.e., description, assessment, or prediction)?**
  
  Considering the classification proposed by Wagner (2013), we classified the studies S04, S17 and S18 as definition quality models since the authors used them to propose a set of models for meeting the system’s non-functional requirements, to guide architectural activities and to show relationships between quality attributes and design aspects, respectively. Studies S10, and S22 were classified as assessment models, since both established quality models that were used to support architectural evaluations.

- **RQ1.3: Which elements comprise the QM?**
  
  In S04, the quality model was built following the rules of a metamodel proposed by the authors. Such metamodel, essentially presents two types of elements: non-functional requirements and properties. Non-functional requirements refer to quality requirements such as comfort, stability, and safety and are on top of the model. Likewise, properties refer to a set of behavioral attributes of a system, the environment in which it is operating and its users. For instance, in the automotive domain, examples of properties may be vehicle stability, acceleration, and control accuracy. Properties may be linked to non-functional requirements or to other properties.

  In S10, the quality model was built following the rules of the QADAG metamodel. Such metamodel comprises nine elements: quality attribute, leafQA, compositeQA, Evaluation-Technique, Joining Technique, DataType, Scenario, Constraint, and Interpretation. The structure of quality attributes is based on the composite pattern, therefore, a QA can have two specializations: leafQA, which contains an evaluation technique that describes how to evaluate the architecture regarding that attribute, and compositeQA, which contains a joining technique that provides instructions on how to combine many evaluation results
into one (aggregation). Constraints can be associated to many quality attributes and vice-versa (many-to-many relationship). Restrictions on the technology to be used or vendors are examples of constraints. Scenarios also have a many-to-many relationship with quality attributes and have the same meaning of the scenarios described within Section 2.3.1. DataType represents the value resulted from an EvaluationTechnique or a JoiningTechnique, e.g., resource utilization in % or costs per unit. Interpretation results from the analysis of results from evaluations regarding an attribute in light of stated requirements, aiming at facilitating communication among stakeholders to support design decisions.

In S17, quality attributes were divided into three categories: cost attributes (e.g., development costs, product costs and maintenance costs), product attributes (energy usage, communication performance, and dependability) and delivery process attributes (e.g., development feasibility, and manufacturing and service feasibility). Some of these attributes were considered sub-characteristics and further divided into more attributes.

In S18, authors address two levels within their quality model. A set of quality attributes is defined and each attribute linked to a design aspect. Quality attributes represent the problem space, i.e., requirements that must be satisfied. On the other hand, design aspects such as error handling, initialization, and testing represent the solution space.

In S22, authors defined their model in basically two levels of abstraction. Major criteria, also named by the authors by quality characteristics, forms the first and most abstract level of the model. With only one exception, each characteristic is composed by many minor criteria (or quality attributes). These quality attributes represent the second abstraction levels. While that major criteria can be divided into minor criteria, minor criteria are atomic and directly measurable by metrics.

- RQ1.4: How has been the QM evaluated (e.g., using prototypes or software systems)?

None of the quality models found had their structure or quality attributes evaluated regarding completeness or correctness. This is due to their role within the studies we analyzed, which is to support design and evaluation activities, regarding software architecture.

RQ2: What are the most important QA for CES?

From all studies considered, we found a total of 84 QAs reported, counting overlaps. However, as most of them were not standardized regarding their name and definition (i.e., there is an wide variety of terminology used for the same concept by different authors), we performed a procedure to map each QA into the characteristics and sub-characteristics of ISO/IEC 25010. The procedure consisted of a comparison between each one of the 84 QAs against the characteristics and sub-characteristics of ISO/IEC 9126-1 and 25010 (ISO/IEC 9126, 2001; ISO/IEC 25010, 2011), and ISO/IEC/IEEE 24765 (ISO/IEC/IEEE 24765., 2010). We classified the 84 QAs in two groups: (1) QAs reported and defined by the authors of the study; and (2) QAs reported
Table 3 – Information summary for RQ1

| Study Model | Quality Model | Definition Approach | Purpose | Composition Elements | Evaluation Approach |
|-------------|---------------|---------------------|---------|----------------------|---------------------|
| S04         | Metamodel     | Definition          | NFR, Properties | None                 |
| S10         | QADAG         | Assessment          | Quality attributes, leafQA, compositeQA, EvaluationTechnique, Joining Technique, DataType, Scenario, Constraint, and Interpretation | None |
| S17         | None          | Definition          | Quality attributes, categories | None |
| S18         | None          | Definition          | Quality attributes, design aspects | None |
| S22         | Tailored classic QMs | Assessment | Quality characteristics, quality attributes | None |

without any definition presented by the authors of primary studies. For each group, we followed different steps, as follows.

- **For quality attributes presented by studies along with a definition (group 1):** Firstly, we compared the definition of each attribute found in our systematic mapping to the definition of the 53 quality attributes of ISO/IEC 25010. If the definition matches, we then mapped the quality attribute into the quality characteristic of the standard, i.e., we consider the characteristic in place of the attribute. If we did not find a match within the ISO 25010, then we performed the same comparison against the 44 quality attributes of ISO/IEC 9126-1. Finally, if we did not find any matches, we consider the attribute as new candidate for our quality model.

- **For quality attributes presented by studies without a definition (group 2):** Firstly, we searched for the quality attribute name at ISO 25010. If we found a match, then we directly map the attribute into the characteristic of ISO 25010. The same is done to ISO 9126-1 if we did not find a match in ISO 25010. After that, if we did not find a match in both standards, then we search the quality attribute name within IEEE 24765. If we find a match, then we pick its definition and try to map it considering the procedure for quality attributes with definitions.
The overall procedure is shown in Figure 16.

After mapping the QAs into the characteristics and sub-characteristics of the ISO/IEC 25010, we calculated their frequency to figure out which ones are more common and, therefore, more likely to be found in the field of CES. It is also an indicative of what characteristics are more important. Reliability was the most frequent characteristic (9), followed by time behaviour (8), resource utilization (7), availability (5), health and safety risk mitigation (5), adaptability (4),
3.2. Systematic Literature Mapping

Fault tolerance (4), maintainability (4), performance efficiency (4), integrity (3), interoperability (3), modularity (3), freedom from risk (3), economic risk mitigation (2), portability (2), testability (2), analysability (1), capacity (1), coexistence (1), comfort (1), compatibility (1), confidentiality (1), environmental risk mitigation (1), functional completeness (1), functional suitability (1), installability (1), integrity (1), modifiability (1), recoverability (1), reusability (1), and security (1). Figure 17 shows the frequency of these characteristics and sub-characteristics.

![Figure 17 – Frequency of Quality Characteristics](image)

**RQ2.1: Which are the QA established for the different CES sub-domains?**

We found nine application domains within the analyzed: nuclear (1), transportation (2), industrial systems (1), robotics (1), military (2), railways (1), avionics (2), healthcare (2), automotive (9). One study did not reported any application domain. Table 4 shows the list of quality attributes for each domain identified in this SM.

**RQ2.2: How the QA were defined (e.g., using domain-specific or generic QM, or based on accumulated experience)?**

Most studies did not present a definition for QAs addressed (S01, S02, S03, S04, S07, S10, S11, S13, S14, S19, S20).

In S05, the authors address three QAs: performance, reliability and energy consumption. They are indirectly defined when the authors discuss their approach to evaluate these attributes. For performance, they cited latency, throughput, and utilization metrics. For reliability, component reliability (CHEUNG et al., 2008) (i.e., proportion of time spent in normal operation mode) was used, and for energy consumption, they mentioned the energy consumption estimation...
Table 4 – Quality attributes organized by domain

| Application domain | Quality attributes                                                                 |
|--------------------|-------------------------------------------------------------------------------------|
| Automotive         | Adaptability, availability, capacity, co-existence, comfort, compatibility, fault  |
|                    | tolerance, freedom for risk, functional completeness, health and safety risk        |
|                    | mitigation, installability, integrity, interoperability, maintainability, mainta-   |
|                    | inability, modularity, performance efficiency, portability, recoverability,        |
|                    | reliability, resource utilization, reusability, testability, and time behaviour    |
| Avionics           | Reliability, resource utilization, security, testability, and time behaviour       |
| Industrial         | Adaptability, economic risk mitigation, health and safety risk mitigation,         |
|                    | maintainability, and time behaviour                                               |
| Healthcare         | Analysability, availability, freedom from risk, functional suitability, modifi-     |
|                    | ability, modularity, reliability, resource utilization, and time behaviour        |
| Military           | Adaptability, economic risk mitigation, fault tolerance, health and safety risk    |
|                    | mitigation, interoperability, and modularity                                       |
| Railways           | Fault tolerance                                                                    |
| Robotics           | Performance efficiency, reliability, and resource utilization                     |
| Transportation     | Maintainability, performance efficiency, portability, and time behaviour         |
| Nuclear            | Performance efficiency                                                             |

technique (SEO; MALEK; MEDVIDOVIC, 2008), which calculates energy consumption based on platform- and application-specific parameters (communication among software and devices), and CPU and memory usage.

S06 addresses testability and the authors define by their own as “the ease of testing a piece of software design using structural testing strategies”.

In S08, authors provide a definition for the attribute safety based on a reference: “freedom from accidents and loss”. However, for dependability, a set of attributes (availability, reliability, safety, confidentiality, integrity, and maintainability) is given to support its definition. However, attributes within this set are not defined.

In S09 the quality attributes addressed are defined as requirements that the software platform must fulfill. Authors discuss composability, deployability, stability, configurability, and dependability.

In S12, authors address the fault tolerance, and they define it describing what is a fault tolerant component: a component is fault tolerant if it can avoid service failures in presence of errors, which means that the component provides an acceptable (full or degraded) level of operation.

Authors of S15 use the definition of (BARBACCI et al., 1995) and (AVIŽIENIS et al., 2004) for the following attributes: dependability, availability, reliability, integrity, confidentiality, safety, and maintainability.

In S16, three attributes are addressed and defined by the authors. Performance was
defined as the time it takes for a communication component to transfer one message originated from one process or thread to another. Maintainability was defined by the maintainability index metric (PEARSE; OMAN, 1995) applied to components. Finally, portability was defined as the effort needed to move prototypes and communication components among different platforms such as windows and linux.

The study S17 presents a QM as discussed in RQ1, with a total of 31 attributes. Many of them are not defined by the authors although their definition can be easily inferred. The other attributes that are not common, have a brief explanation about their meaning, as is the case of the attribute commercial efficiency, defined as “how well the solution matches what suppliers can offer”.

In S18 the author does not present the definition for QAs, but references a previous work ((WIJNSTRA, 2001)) that contains the definitions for attributes addressed in S18.

S21 addresses three attributes, energy consumption, availability and performance. Except for energy consumption, which was not defined, the authors define availability and performance as requirements for the system used as example in the study, a pacemaker. Performance, for example was defined as “Performance refers to the capacity of the system to control heartbeats according to the timing requirements given in the Pacemaker specification.”.

Authors of S22 provide an exclusive section in which they introduce and define attributes contained in their model. The authors themselves highlight that some attributes are defined in a general manner and other are described (not only defined) in an extensive manner and specialized to the automotive domain.

Finally, in S23, authors address the following attributes: response time, processor utilization, communication line utilization, system safety, and system cost. Response time, processor utilization, communication line utilization, and system safety were defined as possible metrics that could be used to measure the attributes themselves. System cost was not defined but highlighted as an important attribute.

**RQ3: Which metrics have been used to measure the QA?**

In S01, authors use the UML-MAST to model two candidate architectures and measure their performance (time behaviour) in two scenarios. To do this, they measure the response time in milliseconds for a set of transactions within each scenario. The sum of response times was used to define which architecture better fits the stated requirements and support project decisions.

In S05, five metrics are discussed. For performance efficiency, simulation using Layered Queuing Network (LQN) enables the application of three metrics: latency, throughput, and utilization. These metrics are discussed in Rolia and Sevcik (1995). Simulation of Hidden Markov Model (HMM) is used to calculate component reliability. Resource utilization is calculated considering an energy consumption estimation technique proposed by Seo, Malek and Medvidovic (2008).
The author of S06 uses two metrics to support testability measurement in data flow designs: controllability and observability.

In S22, the authors develop 8 metrics to measure three attributes contained in their quality model and then evaluate software architectures in driver assistance systems: (i) structural understandability, (ii) component understandability, (iii) changeability, (iv) stability, and (v) testability are computed to measure Mutability; (vi) scalability and (vii) resource efficiency support the measurement of Efficiency; and (viii) modularity measures Reusability;

RQ3.1: Which metrics were established for the QA?

Considering the mapping procedure of quality attributes found in this SM into ISO/IEC 25010 characteristics, we have the following metrics:

Performance efficiency (Capacity): throughput (S05)
Performance efficiency (Time behaviour): response time (S01), latency (S05)
Performance efficiency (Resource utilization): energy consumption estimation technique (S05), utilization (S05), resource efficiency (S22)
Reliability (Maturity): component reliability (S05)
Maintainability (Testability): controllability (S06), observability (S06), and testability (S22).
Maintainability (Modifiability): changeability (S22)
Maintainability (Modularity): stability (S22)
Maintainability (Analyzeability): structural understandability (S22), component understandability (S22)
Portability (Adaptability): scalability (S22)

RQ3.2: How these metrics were defined?

In general, metrics are defined by authors or have their definition taken from another study. In S01, for instance, authors do not provide a definition for the metric used (i.e., response time), as it is implicitly implemented in the tool used for simulation and performance evaluation. In S05, metrics were briefly defined and the source studies from which they were taken were referenced by authors. In S06, definitions for metrics were given by the own author. Finally, in S22, all eight metrics were also defined by authors.

RQ3.3: How have been the metrics evaluated (e.g., using prototypes or real systems)?

None of the studies have conducted an activity or followed a systematic approach to evaluate the metrics used within their studies. This is probably due to the main goal of their studies, which were commonly related to the evaluation of QAs in software design.
3.2.3 Discussion of results

In this section, synthesis of results of our SM are presented.

3.2.3.1 QM for CES

Quality analysis of CES has been a focus of study in recent years, since included studies in this SM were published in the last 15 years. Despite the importance of developing high-quality CES, few works addressing assessment and none work investigating prediction of CES quality were found. Hence, quality of CES has been focused on describing non-functional requirements and properties for those systems.

Three QMs found in this SM were constructed based on meta-models or well-known standards, which have high-level of abstraction. In this context, future QMs for CES in different application domains can be developed based on pre-defined models by means of instantiation of common QAs for CES in the desired domain.

As detailed in Table 3, only for two application domains, i.e., automotive and healthcare, QM have been proposed. In this perspective, more efforts must be done to provide models to analyze quality in other domains such as avionics, military, robotics, and such presented in Table 4. Moreover, the first QM for avionics was proposed in the year 2009, and the last one in 2013. However, analyzing other included primary studies we identified that the other 6 studies proposing QAs for automotive systems have not used any of the established QMs for such domain. Similarly was in the context of healthcare. In this perspective, formalization, maturity, standardization and more dissemination of such QMs can be done to orient the quality analysis of those domains.

3.2.3.2 QAs for CES

As presented in Figure 17, we found 31 QAs considered important by primary studies for CES development. Such QAs have been proposed based on evidence obtained from application domains, such as, automotive, avionics, industrial, healthcare, military, railways, robotics, transportation, and nuclear systems.

From Table 4 and Figure 17, we can infer that reliability, time behaviour, resource utilization, and performance efficiency, besides be one of the QAs most found in this SM. They are of interest of many application domains, as depicted in Table 5.

| QA                        | Domains                                      |
|----------------------------|----------------------------------------------|
| Reliability                | Automotive, avionics, healthcare, and robotics |
| Time behaviour             | Automotive, avionics, industrial, healthcare, and transportation |
| Resource utilization       | Automotive, avionics, healthcare, and robotics |
| Performance efficiency     | Automotive, robotics, transportation, and nuclear |
Considering the QMs for automotive (S04, S10, S17, and S22) and healthcare (S18) systems, we identify such models do not represent all QAs for the related domains. For instance, for automotive any of the 3 QMs found established co-existence and comfort as important QAs, however, in our SM we found that such QAs are important for automotive systems (see Table 4). Regarding healthcare systems, time behaviour, resource utilization and availability were not considered by the QM in S18.

In this perspective, results of this SM can be used to create new QMs for specific application domains, to evolve existing QMs and to guide the selection of QAs for CES in domains where no QMs have been proposed.
QUALITY MODEL FOR CRITICAL EMBEDDED SYSTEMS (QM4CES)

The QM herein proposed has the main objective to support architectural activities throughout the development of CES. Besides that, as QMs usually capture relevant aspects regarding what quality is in a given context, it is possible to use this QM beyond architectural scope, specially to support other software engineering activities that address system qualities. For instance, it can be useful to specify requirements of CES, to serve as baseline for the establishment and application of metrics, to support evaluation of software products of CES, and as an initial reference for constructing specific QMs for organizations, projects, and phases of projects. As we do not expect to cover specific applications domains in which CES have been employed, the level of abstraction of our QM lies between general QMs, such as the proposed by ISO/IEC 25010 (2011), Dromey (1995), Boehm, Brown and Lipow (1976), McCall, Richards and Walters (1977), Lochmann and Goeb (2011)) and domain-oriented QMs (e.g., the proposed by Franch et al. (2008), Behkamal, Kahani and Akbari (2009), Villalba, Fernández-Sanz and Martínez (2010), Kim and Lee (2009). Hence, our QM is composed of a subset quality characteristics of ISO/IEC 25010 (2011), as QMs must be reduced to eliminate redundancy and increase its efficiency (VILLALBA; FERNÁNDEZ-SANZ; MARTÍNEZ, 2010). Therefore, it is not oriented towards only one domain, but a set of systems that share characteristics of CES and are spread among many application domains, such as avionics, automotive, and healthcare.

Considering the work of Wagner (2013), our QM can be classified as a definition model, as it presents a set of quality characteristics, their decomposition into sub-characteristics, their definitions (which are adapted to the context of CES), and trade-off relations between quality attributes. For Kläsi, Lampasona and Münch (2011), our QM is also considered a public-level model, since it can be useful for general use (quality evaluation and requirements specifications) or for use in a specific domain. Moreover, it can be tailored for specific purposes, and it is useful during software engineering activities, showing adherence to the definitions and organization of
Chapter 4. Quality Model for Critical Embedded Systems (QM4CES)

4.1 Phase 1 - Initial Analysis

In this phase, it is necessary to clearly define the field for which the model will be constructed and, based on it, a set of characteristics from the standard ISO/IEC 25010 (2011) is selected and adapted to fulfill the field particularities. When DUMOD was published, the ISO/IEC 9126 (2001) standard was the main reference standard of general QMs. In the same year, this standard was superseded by ISO/IEC 25010 (2011), which brought some changes to the QM previously proposed. In this sense, we used the ISO/IEC 25010 (2011) for adaptation instead of the ISO/IEC 9126 (2001).
4.1. Phase 1 - Initial Analysis

4.1.1 Definition and analysis of the field

The field of CES counts on a series of systems already employed by industry within their main products (cars, airplanes, etc.). In fact, to be part of this field, a system must present characteristics of embedded systems concerning its hardware and software components as discussed in Chapter 2. Moreover, a system is considered CES if, in case of failures or unintended behavior, the integrity of the environment, human beings, and products in which it is embedded or it is interacting with (specially valuable ones) are put in risk of harm or destruction. In order to establish and understand the bounds of this field, we designed a conceptual model (see Figure 19) relating concepts derived from CES definitions (found in Bate (2008), Aguiar et al. (2010)). A CES is commonly embedded into a larger engineered product, being part of it or perform a critical function for it (e.g., it might not be sharing the same infrastructure or hardware platform of a product).

An engineered product can be, for example, one of the many types of vehicles, such as cars, trains, and airplanes. Considering, for example, cars, most systems are embedded into them, while that traffic light control systems are not part of cars, but perform a critical function for traffic control and car safety. In turn, some engineered products are used to physically interact with valuable products, which are mainly the case of many industrial systems used to build and/or move manufactured products. In the same sense, some engineered products of the healthcare domain interact directly with human beings for medical purposes. This interaction can be invasive (e.g., infusion pumps) or non-invasive (e.g., ultrasound devices). Additionally, many engineered systems, specially vehicles, can be used for logistics purposes, carrying human beings and/or valuable cargo for long distances.

Finally, engineered products operate in environments with human beings and valuable products. This environment is, in many cases, unpredictable regarding physical aspects due to interference of natural phenomena. Any failure in engineered products or CES may lead to different types of damage, such as electrocution, kinetic, physical and heat damage to products and human beings, air pollution and water contamination in case of release of chemical compounds to air and water environments.

4.1.2 Adaptation of ISO/IEC 25010 standard to CES field

Based on both models (product and quality in use) presented by ISO/IEC 25010 (2011) and evidence found in the results of the SM presented in Chapter 3, we selected a set of characteristics and sub-characteristics we consider important regarding CES. The quality characteristics and sub-characteristics from ISO/IEC 25010 (2011) and their definition are shown in Tables 6 (product quality) and 7 (quality in use). Besides that, these tables pointed out which characteristics and sub-characteristics were selected to compose our QM.

Next, we discuss all characteristics and their sub-characteristics presented in Tables 6
Figure 19 – CES concept map
| Characteristic            | Definition                                                                 | Selected |
|--------------------------|---------------------------------------------------------------------------|----------|
| Functional suitability   | Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions |          |
| Functional completeness  | Degree to which the set of functions covers all the specified tasks and user objectives |          |
| Functional correctness   | Degree to which a product or system provides the correct results with the needed degree of precision |          |
| Functional appropriateness | Degree to which the functions facilitate the accomplishment of specified tasks and objectives |          |
| Reliability              | Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time | ✓        |
| Maturity                 | Degree to which a system, product or component meets needs for reliability under normal operation | ✓        |
| Availability             | Degree to which a system, product or component is operational and accessible when required for use | ✓        |
| Fault Tolerance          | Degree to which a system, product or component operates as intended despite the presence of hardware or software faults | ✓        |
| Recoverability           | Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system | ✓        |
| Performance Efficiency   | Performance relative to the amount of resources used under stated conditions | ✓        |
| Time behaviour           | Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements | ✓        |
| Quality Attribute          | Definition                                                                 |  |
|---------------------------|---------------------------------------------------------------------------|---|
| Resource utilization      | Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements | ✓ |
| Capacity                  | Degree to which the maximum limits of a product or system parameter meet requirements | ✓ |
| **Usability**             | Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use | ✓ |
| Appropriateness recognisability | Degree to which users can recognize whether a product or system is appropriate for their needs |   |
| Learnability              | Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use | ✓ |
| Operability                | Degree to which a product or system has attributes that make it easy to operate and control | ✓ |
| User error protection     | Degree to which a system protects users against making errors              | ✓ |
| User interface aesthetics | Degree to which a user interface enables pleasing and satisfying interaction for the user |   |
| Accessibility             | Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use |   |
| **Maintainability**       | Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers |   |
| Modularity                | Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components | ✓ |
| Reusability               | Degree to which an asset can be used in more than one system, or in building other assets |   |
| Characteristics | Definition                                                                                                                                                                                                 | ✓  |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Analysability   | Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified                  |    |
| Modifiability   | Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality                                                                   | ✓  |
| Testability     | Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met                                | ✓  |
| Security        | Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization                        | ✓  |
| Confidentiality | Degree to which a product or system ensures that data are accessible only to those authorized to have access                                                                                             | ✓  |
| Integrity       | Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data                                                                               | ✓  |
| Non-repudiation | Degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later                                                                                | ✓  |
| Accountability  | Degree to which the actions of an entity can be traced uniquely to the entity                                                                                                                              | ✓  |
| Authenticity    | Degree to which the identity of a subject or resource can be proved to be the one claimed                                                                                                                     | ✓  |
| Compatibility   | Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment                     | ✓  |
| Co-existence    | Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product                              | ✓  |
| Term          | Definition                                                                                   |
|--------------|--------------------------------------------------------------------------------------------|
| Interoperability | Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged |
| Portability  | Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another |
| Adaptability | Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments |
| Installability | Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment |
| Replaceability | Degree to which a product can replace another specified software product for the same purpose in the same environment |
4.1. Phase 1 - Initial Analysis

Table 7 – Quality in use characteristics from ISO/IEC 25010, definitions, and selection flag

| Characteristic | Definition | Selected |
|----------------|------------|----------|
| **Satisfaction** | Degree to which user needs are satisfied when a product or system is used in a specified context of use |          |
| Usefulness | Degree to which a user is satisfied with their perceived achievement of pragmatic goals, including the results of use and the consequences of use |          |
| Trust | Degree to which a user or other stakeholder has confidence that a product or system will behave as intended | ✓ |
| Pleasure | Degree to which a user obtains pleasure from fulfilling their personal needs |          |
| Comfort | Degree to which the user is satisfied with physical comfort |          |
| **Effectiveness** | accuracy and completeness with which users achieve specified goals |          |
| Freedom from risk | Degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment | ✓ |
| Economic risk mitigation | Degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use | ✓ |
| Health and safety risk mitigation | Degree to which a product or system mitigates the potential risk to people in the intended contexts of use | ✓ |
| Environmental risk mitigation | Degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use | ✓ |
| **Efficiency Context Coverage** | resources expended in relation to the accuracy and completeness with which users achieve goals |          |
| Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified |          |
| Context completeness | Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Flexibility         | Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements |
and 7 together with the rationale behind their selection or exclusion for our QM, and a list of studies found in the SM presented in Chapter 3 that address the corresponding characteristic.

- **Functional Suitability**
  - **Functional completeness (not selected)**: This characteristic is very important for CES, as the lack of crucial functions may invalidate the entire system. Our focus is on software architecture, which means that non-functional aspects are more relevant than functional aspects. This way we choose not to pick this characteristic. Studies: Wijnstra (2003) and Ahrens et al. (2013).
  - **Functional correctness (not selected)**: Miscalculations are not welcome for any kind of software system. For CES, it may lead to wrong or less-than-optimal execution. Similarly, this characteristic does not emphasize architectural concerns, hence, we decided not to select it. Studies: Wijnstra (2003).
  - **Functional appropriateness (not selected)**: The way functionalities support users to perform tasks and achieve their objectives is also secondary to the architecture. Studies: Wijnstra (2003).

- **Reliability**
  - **Maturity (selected)**: This characteristic refers to the capability of the system to be available under normal operation. That means all system components (hardware and software) are working as intended and the system is fully operational with the absence of failures. For CES, this characteristic is essential. Studies: Meedeniya et al. (2010), Enard et al. (2013), Meedeniya, Aleti and Moser (2015), Al-Daajeh, Al-Qutaish and Al-Qirem (2011), Wijnstra (2003), and Insaurralde et al. (2008).
  - **Availability (selected)**: CES must be always ready to perform its functions for the engineered product it is embedded into. Actually, many systems operate at least in passive mode, gathering data from the environment through sensors that will eventually trigger an active mode in which real-time operations can take place. Studies: Meedeniya et al. (2010), Enard et al. (2013), Meedeniya, Aleti and Moser (2015), Al-Daajeh, Al-Qutaish and Al-Qirem (2011), Axelsson (2009), Wijnstra (2003), Insaurralde et al. (2008), Loniewski et al. (2013), and Ahrens et al. (2013).
  - **Fault Tolerance (selected)**: As CES perform critical functions for engineered products, and in case of hardware or software failure, the systems must recover themselves or at least enter in fail-safe state and warn somehow the potential interested entities responsible for fixing/solving the problem or just be aware of the failure. This characteristic is one of those that drive the development of high-integrity real-time systems (HIRTS) (RAĐENOVIC; PAIGE, 2006). Studies: Meedeniya et al. (2010), Xu et al. (2012), Enard et al. (2013), Meedeniya, Aleti and Moser (2015), Al-Daajeh,
-- Recoverability (selected): In case of failure, it is desirable that CES try to recover themselves to their normal operation state. If it is not possible, the systems should present some mechanism for a contingency plan. Studies: Meedeniya et al. (2010), Enard et al. (2013), Meedeniya, Aleti and Moser (2015), Al-Daajeh, Al-Qutaish and Al-Qirem (2011), Wijnstra (2003), Insaurralde et al. (2008), and Ahrens et al. (2013).

- Performance Efficiency

- Time behaviour (selected): CES impose sometimes real-time constraints. In many cases, these constraints are hard-real time, which means that if not achieved, the system failed to perform its functions, presenting a risk. A timing requirement was evaluated using ATAM in Ortiz et al. (2003) to check if a reference architecture could be reused in the context of a teleoperation system. Benyahia, Potvin and Xu (2006) evaluates the impact of changes in system’s environment to timing requirements, due to the importance of such attribute for customer satisfaction. Methods for interaction and communication in CNC systems must meet strict timing requirements (Du et al., 2009). CNC systems are largely employed by many industries of any scale. Studies: Ortiz et al. (2003), Benyahia, Potvin and Xu (2006), Du et al. (2009), Enard et al. (2013), Martensson, Grahn and Mattsson (2005), Axelsson (2009), Insaurralde et al. (2008), Loniewski et al. (2013), Ahrens et al. (2013), and Etemaadi et al. (2013).

- Resource utilization (selected): In many cases, CES are embedded into engineered products (e.g., aircrafts and motor vehicles), which have energy sources hard to recharge (Meedeniya et al., 2010). System’s longevity is positively impacted when software energy consumption is reduced (Malek et al., 2010). Moreover, such sources are sometimes shared with other systems embedded into the same engineered products. Low resource utilization is desired to increase system’s availability. In nano devices, which require the minimal size of batteries, this characteristic is even more important to be optimized (Meedeniya et al., 2010). Studies: Meedeniya et al. (2010), Malek et al. (2010), Axelsson (2009), Insaurralde et al. (2008), Loniewski et al. (2013), Ahrens et al. (2013), and Etemaadi et al. (2013).

- Capacity (selected): Despite of the limits presented by the system (e.g., the number of concurrent users, communication bandwidth, throughput of transactions, and size of database), it must be able to meet explicit requirements. Studies: Axelsson (2009).

- Usability

- Appropriateness recognisability (not selected): Actually, CES are fundamental for engineered systems to replace mechanisms previously employed to perform critical functions, representing a technological advance aiming at improving the quality of
engineered products. However, the presence of CES in such engineered products is not always recognizable by users. Therefore, we choose not to pick this characteristic.

- **Learnability (selected)**: Users of the system must be able to understand it for appropriate operation. For CES, interactions between the users and system affect the behavior of the whole engineered product. Therefore, learnability enables users to efficiently learn how to use the system, which impacts on the interaction between users and system during normal operation and emergency situations. We did not find evidence of this characteristic in the SM, but we choose to include it since it is related to usability. Although usability seems to be unrelated to software architecture, Bass and John (2001) and Vilela et al. (2015) show the contrary.

- **Operability (selected)**: CES must be easy to operate by their users. We selected this sub-characteristic, as it can decrease the probability of user errors. We did not find evidence of this characteristic in the SM, but we choose to include it since it is related to usability. Although usability seems to be unrelated to software architecture, Bass and John (2001) and Vilela et al. (2015) show the contrary.

- **User error protection (selected)**: User errors during system operation can be catastrophic. Therefore, CES must be aware of its state to protect itself against users requesting it to perform risky operations. Moreover, CES must act in case of absence or inactivity of users in a risky scenario. We did not find evidence of this characteristic in the SM, but we choose to include it since it is related to usability. Although usability seems to be unrelated to software architecture, Bass and John (2001) and Vilela et al. (2015) show the contrary.

- **User interface aesthetics (not selected)**: Although this characteristic is important for some application domains, we believe it is not representative for the CES field.

- **Accessibility (not selected)**: Although this characteristic is very important for some application domains of CES such as healthcare, we understand that it is a particularity that do not affect most CES.

**Maintainability**

- **Modularity (selected)**: Modularity was selected as it represents the degree to which CES is composed of discrete (i.e., well defined) components. In turn, changes in discrete components have minimal impact on other components. This is therefore useful to avoid introduction of errors during inevitable system maintenance. Studies: Wijnstra (2003), Durisic et al. (2013), and Ahrens et al. (2013).

- **Reusability (not selected)**: Reusability is desired to decrease system’s cost and increase its quality by the use of artifacts previously employed for system’s development. However, reusable assets require more time for development as they must be generic enough for reuse in many contexts. We choose not to select this characteristic.
as it is not directly related to the main characteristics of CES. Studies: Durisic et al. (2013) and Ahrens et al. (2013).

- **Analysability (selected)**: When change requests emerge, it is important to assess its impacts on the system before maintenance begins. It is crucial for CES considering its critical aspects. Studies: Wijnstra (2003) and Durisic et al. (2013).

- **Modifiability (selected)**: Modifiability concerns the degree to which the system can be modified without introducing defects or degrading the system’s overall quality. For CES, defects can lead to failures that can lead to undesired outcomes. Studies: Al-Daajeh, Al-Qutaish and Al-Qirem (2011), Martensson, Grahn and Mattsson (2005), Wijnstra (2003), and Durisic et al. (2013).

- **Testability (selected)**: A CES must be tested before deploying it to the use. Testability is therefore very important to ensure an effective and efficient testing policy. Studies: Axelsson (2009), Traon, Ouabdesselam and Robach (2000), and Durisic et al. (2013).

**Security**

- **Confidentiality (selected)**: Unauthorized access puts the system in risk as it may be performed by malicious person intending to cause damage for some reason. Therefore, this characteristic is very important as systems are becoming even more connected than in the past.

- **Integrity (selected)**: CES must provide mechanisms to protect its software and data from unauthorized modification, as it may enable malicious person or software to partially or fully compromise the system. Studies: Al-Daajeh, Al-Qutaish and Al-Qirem (2011), Axelsson (2009), Insaurralde et al. (2008), Insaurralde et al. (2008), and Ahrens et al. (2013).

- **Non-repudiation (selected)**: This characteristic might be very useful for many systems as they can keep track of information about operations performed by the system or users that can be useful for later use. For example, this kind of information can be useful to identify if the source of an accident was human or machine error.

- **Accountability (selected)**: This characteristic complements the other characteristics of security. It must be possible to precisely identify the entities (be it a person or external system) behind an action that might impact the operation of CES.

- **Authenticity (selected)**: In order to achieve the other characteristics, suitable authentication mechanisms should be employed to ensure authenticity of person and software entities. Studies: Al-Daajeh, Al-Qutaish and Al-Qirem (2011) and Insaurralde et al. (2008).

**Compatibility**
4.1. Phase 1 - Initial Analysis

- **Co-existence (selected):** CES can share resources with other embedded systems within engineered products. Moreover, in many cases, CES are part of a system-of-systems (a class of system resulted from interoperability among independent systems that work together to achieve goals that could not be achieved by systems working individually (LUZEAUX; RUAULT, 2010; MAIER, 1998; DOD, 2008)), therefore sharing a common environment and resources with other constituent systems of the system-of-systems. In this sense, CES must be able to perform its functions regardless the current conditions of the environment and resources without compromising their capabilities or other constituents. Studies: Axelsson (2009).

- **Interoperability (selected):** This attribute refers to the ability of two or more systems or components to exchange information and use that information. Information exchange is crucial in many cases to allow systems and components to perform their main functions and, therefore, any flaw during information exchange and interpretation might lead to system failure. Therefore, we opted to include this attribute. Studies: Axelsson (2009) and Ahrens *et al.* (2013).

- **Portability**

  - **Adaptability (selected):** Many CES operate in unpredictable environments that impose restrictions for their operation. For instance, restrictions range from the unavailability of one or more sensors due to unpredicted events or conditions to the entire system. Studies: Martensson, Grahn and Mattsson (2005) and Ahrens *et al.* (2013).

  - **Installability (not selected):** This characteristic might be useful to support the deployment of software into hardware components of CES. However, it is not a major concern for its software architecture.

  - **Replaceability (not selected):** Most CES have long life-cycles. Therefore, the impact of this characteristic for development is minimal compared to the others.

- **Satisfaction**

  - **Usefulness (not selected):** We found no evidence to select this characteristic.

  - **Trust (selected):** Functions performed by CES are very important to guarantee the overall quality of the engineered system in which they are embedded. Trust means the system’s stakeholders are confident about systems and expect they to do what they are supposed to do.

  - **Pleasure (not selected):** We consider this characteristic might be relevant to the engineered system as a whole instead of the CES itself.

  - **Comfort (not selected):** This characteristic is for sure very important for many vehicles but we do not think it is critical or even architecturally significant.
• **Effectiveness** (not selected): We found none evidence to select this characteristic.

• **Freedom for risk**

  – **Economic risk mitigation** (selected): As CES are commonly embedded into engineered systems of high value or handle valuable products or cargo, it is important to ensure risk mitigation in case of failure to avoid harm or destruction of such items. Du et al. (2009) point out this requirement as important to avoid systems break down. Studies: Du et al. (2009), Axelsson (2009), Wijnstra (2003), and Etemaadi et al. (2013).

  – **Health and safety risk mitigation** (selected): CES failures can cause several types of damage to human beings or components of the environment, depending on the scenario and the system in which they are embedded. Any kind of damage is undesired as it may lead to death or unrecoverable harms to human beings. Du et al. (2009) highlight the importance of safety to avoid machine operators from getting hurt or killed. Low failure rates are required to be exhibited by the systems (Du et al., 2009). Studies: Al-Daajeh, Al-Qutaish and Al-Qirem (2011), Du et al. (2009), Axelsson (2009), Wijnstra (2003), Ahrens et al. (2013), and Etemaadi et al. (2013).

  – **Environmental risk mitigation** (selected): CES operate in a large set of environments, ranging from indoor rooms to natural biomes. A failure can compromise the engineered systems they are embedded into. Many of these engineered systems carry cargo that can be set in fire, destroying physical entities and/or polluting the air, can contain chemicals that can pollute water environments, or release debris that if not collected will hurt the environment. Studies: Al-Daajeh, Al-Qutaish and Al-Qirem (2011), Axelsson (2009), Wijnstra (2003), and Etemaadi et al. (2013).

• **Efficiency** (not selected): We found no evidence to select this characteristic.

• **Context Coverage**

  – **Context completeness** (not selected): This characteristic issues that all planned functions that systems have to perform must present a desired level of effectiveness, efficiency, freedom from risk, and satisfaction. From the architectural point-of-view, these four ones are considered separately, therefore, we did not select this characteristic, since we considered each one separately for our model establishment. Additionally, we found no evidence to select this characteristic.

  – **Flexibility** (not selected): This characteristic is important for autonomous systems, but it is not applicable for most CES.
4.2 Phase 2 - Definition of Preliminary Model

In this phase, the initial set of characteristics is defined and revised by internal experts of the research group.

4.2.1 Creation of the initial set of attributes

The selection of characteristics from ISO/IEC 25010 (2011) resulted in the following set of attributes to be used in evaluation phase:

- **Product quality**
  - **Reliability**: Availability, Fault tolerance, Maturity, and Recoverability;
  - **Performance efficiency**: Capacity, Resource utilization, and Time behaviour;
  - **Usability**: Learnability, Operability, and User error protection;
  - **Maintainability**: Analyzability, Modifiability, Modularity, and Testability;
  - **Security**: Authenticity, Confidentiality, Integrity, and Non-repudiation;
  - **Compatibility**: Co-existence and Interoperability; and
  - **Portability**: Adaptability.

- **Quality in use**
  - **Satisfaction**: Trust; and
  - **Freedom from risk**: Economic risk mitigation, Environmental risk mitigation, and Health and safety risk mitigation.

The authors of DUMOD gathered additional attributes from diverse literature, including studies and standards related to the domain addressed in their example. As we conducted a systematic mapping to identify which QAs have been addressed in the context of CES, we did not considered the necessity to search in additional sources.

4.2.2 Internal review of the initial set of attributes

Before stepping into the last phase of DUMOD, we presented the preliminary model to experts of our research group for a previous evaluation regarding the selection of characteristics and the rationale we used to justify their selection or not to composite the model. The considerations of the experts were used to improve the content already presented in Section 4.1.2.
4.3 Final version of QM4CES

In this section, we present the final version of QM4CES. After the conduction of model validation, which is presented in Chapter 5, we improved the preliminary model considering feedback provided by survey participants. One new characteristic was added (Reusability). Reusability was suggested for two reasons, and we realized that only after model validation. Firstly because reuse of software is economically interesting, reducing overall price of CES development. Secondly because software that have been previously employed and proven to be reliable is most likely to preserve its characteristics when carefully employed elsewhere. In the context of software architecture, many artifacts can be reused. Reference architectures play an important role in this sense (NAKAGAWA; OQUENDO; BECKER, 2012). For instance, AUTOSAR\(^1\) is employed in the context of automotive domain, as are Continua\(^2\) and UniversAAL\(^3\) established for healthcare domain.

Table 8 shows the final version of QM4CES. Additionally to the quality characteristics selected, we calculated three relative weights based on mean \((W(\mu))\), median \((W(Md))\) and mode \((W(Mo))\) values for each characteristic. These values can be used to support the prioritization of quality characteristics during architectural methods such as ATAM, DCAR, QAW, and PALM.

\(^{1}\) <https://www.autosar.org/>  
\(^{2}\) <http://www.pchalliance.org/continua-design-guidelines>  
\(^{3}\) <http://www.universaal.org/>
Table 8 – Final version of QM4CES

| Characteristic          | Sub-characteristic         | $W(\mu)$ | $W(Md)$ | $W(Mo)$ |
|-------------------------|----------------------------|----------|---------|---------|
| Reliability             | Availability               | 0.79     | 0.80    | 0.80    |
|                         | Fault tolerance            | 0.83     | 0.90    | 1.00    |
|                         | Maturity                   | 0.81     | 0.80    | 0.80    |
|                         | Recoverability             | 0.79     | 0.80    | 0.80    |
| Performance efficiency  | Capacity                   | 0.74     | 0.80    | 0.80    |
|                         | Resource utilization       | 0.75     | 0.80    | 0.80    |
|                         | Time behaviour             | 0.86     | 0.80    | 1.00    |
| Usability               | Learnability               | 0.73     | 0.80    | 1.00    |
|                         | Operability                | 0.70     | 0.60    | 0.60    |
|                         | User error protection      | 0.79     | 0.80    | 1.00    |
| Maintainability         | Analysability              | 0.68     | 0.60    | 0.60    |
|                         | Modifiability              | 0.77     | 0.80    | 0.80    |
|                         | Modularity                 | 0.79     | 0.80    | 0.80    |
|                         | Reusability                | 0.80     | 0.80    | 0.60    |
|                         | Testability                | 0.74     | 0.80    | 0.80    |
| Security                | Authenticity               | 0.68     | 0.60    | 0.60    |
|                         | Confidentiality            | 0.75     | 0.80    | 1.00    |
|                         | Integrity                  | 0.81     | 0.90    | 1.00    |
|                         | Non-repudiation            | 0.73     | 0.80    | 0.60    |
| Compatibility           | Co-existence               | 0.72     | 0.80    | 0.60    |
|                         | Interoperability           | 0.76     | 0.80    | 0.80    |
| Portability             | Adaptability               | 0.66     | 0.60    | 0.60    |
| Satisfaction            | Trust                      | 0.86     | 0.90    | 1.00    |
| Freedom from risk       | Economic risk mitigation   | 0.74     | 0.80    | 1.00    |
|                         | Environmental risk mitigation | 0.78   | 0.80    | 1.00    |
|                         | Health and safety risk mitigation | 0.86 | 1.00    | 1.00    |
QUALITY MODEL EVALUATION

In this chapter we present the third and last phase of DUMOD, performed in the context of our quality model to validate it regarding its completeness and composition. This phase comprises seven activities, as previously explained in Section 2.3.5: (1) Design of questionnaires - addresses the development of the main artifact that will be used for data collection; (2) Data collection - comprising the distribution of the questionnaire to individuals with experience in the CES field; (3) Data analysis; (4) Preliminary Analysis - deals with descriptive statistics (averages, standard deviations, correlations, etc.) applied to data collected in data collection; (5) Reliability Analysis - verifies if there are any sub-characteristic within a characteristic that can be removed from the model; (6) Predictive Analysis - outputs the initial number of sub-characteristics to be kept in the model and the weights of each sub-characteristic inside the model; and (7) Confirmative Validation - assess the validity of the final model. The input for this phase is the list of quality characteristics of the preliminary model obtained after the internal review. Each section of this chapter addresses one of these seven steps from DUMOD model validation phase. The first step, design of questionnaires, was improved with the process proposed by Kasunic (2005) to design effective surveys in software engineering. The remainder of this chapter exhibits each step of the evaluation phase.

5.1 Design of questionnaires

The questionnaire have the main goal of collecting data from our target audience to be used in data analysis and improve our quality model. Main information we are interested are the relevance of each characteristic from different individuals, any characteristic that we missed when building our model, trade-off relations among quality characteristics, and their perceived utility for the quality model being built. To support the development of the questionnaire, we followed the process proposed by Kasunic (2005). Additionally to the goal of this step inside DUMOD approach, that process helped us identify additional goals not addressed by DUMOD,
besides providing a systematic way to execute the survey and therefore improve its general quality.

The process proposed by Kasunic (2005) comprises seven steps, as showed in Figure 20: (1) Identify research objectives; (2) Identify and characterize target audience; (3) Design sampling plan; (4) Design & write questionnaire; (5) Pilot test questionnaire; (6) Distribute the questionnaire; and (7) Analyze results and write a report. Each step is briefly described below:

**Step 1 - Identify research objectives**: In this step the objectives of the survey must be established to guide and scope the survey. The objectives are derived from a problem or issue and provide scope for internal questions.

**Step 2 - Identify & characterize target audience**: Considering the established objectives, it is necessary to define who are the individuals that can give the answers that will support us achieve the stated goals.

**Step 3 - Design sampling plan**: In this step the approach to select individuals that will participate to the survey is defined.

**Step 4 - Design & write questionnaire**: The survey objectives are translated into questionnaire items developed to enable analysis and interpretation.

**Step 5 - Pilot test questionnaire**: The questionnaire is tested by members of target audience to uncover eventual problems and identify improvements.

**Step 6 - Distribute the questionnaire**: The questionnaire is distributed to the target audience; and

**Step 7 - Analyze results and write report**: The results should are collected and presented with easily understandable graphical displays. Reports can be made to support interpretations, inferences, generalizations, and caveats based on evidence found in the results.

![Figure 20 – Survey Process (KASUNIC, 2005)](image)

**5.1.1 Identify research objectives**

In order to establish the objectives of the survey we used the Goal-Question-Metric approach proposed by Basili (1992). This approach is useful when you want to improve something
## 5.1. Design of questionnaires

### Table 9 – Goals, questions and metrics for survey

| Goal | Question | Metric |
|------|----------|--------|
| **G1. Evaluate the completeness of the Preliminary Model** | 1. Does the preliminary model contains every relevant characteristic to describe the quality of CES? | 1. Characteristic weighting  
2. Characteristics added/removed |
| **G2. Establish trade-off relations between [Sub-]Characteristics** | 1. Which characteristics negatively impacts other characteristics? | 2.1 Set of trade-off relations between characteristics |
| **G3. Understand the usefulness of the Quality Model** | 1. Do practitioners believe the quality model - when finished - will be useful for CES?  
2. Which software engineering activities can be supported by the model? | 1.1 Yes/No  
2.1 List of activities |

and was initially proposed for software engineering topics such as processes and methods. It is composed by three main concepts (goal, question and metric) developed in a top-down fashion, starting from goal and finishing at metrics. A goal is composed by a purpose that means what we want to do (e.g., improve an artifact, reduce costs, increase user satisfaction), an issue that is the quality aspect we want to impact (e.g., completeness, performance, reliability), the object that will be improved, and the viewpoint adopted. A goal have one or more questions that help understand how that goal will be assessed or achieved. Metrics are finally used to quantitatively answer each question established for a goal. They can be objective (depends only on the object being measured and does not change from different points-of-view) or subjective (depends both on the object being measured and the point-of-view adopted).

For our survey, we identified three objectives as shown in Table 9. Goals 1 and 2 represent an improvement of the preliminary model in relation to its completeness and soundness. Goal 3 was identified to help us understand the applicability of the model when finished. Goal 1 and 3 were taken from DUMOD approach and are part of the evaluation phase. Goal 2 was established to address software architecture as we expect to identify relations between quality characteristics that are very useful during architectural activities.

### 5.1.2 Identify & characterize target audience

Aiming to address our objectives, we consider the ideal target audience for this questionnaire should be composed by individuals with experience in researching or developing software architectures (e.g., analysis, synthesis and evaluation) for CES. However, to increase the pool of respondents, we will also include individuals with experience designing and developing software (not exactly software architectures) for CES. Members of these two groups (architects and developers) will be identified in the survey to enable us to analyze them separately. The developers
group is probably greater than the first, may not be ideal for our objectives, but can still be useful to uncover new quality attributes that they have considered during CES development.

5.1.3 Design sampling plan

Design sampling deals with how to select a sample from the population of individuals that match our target audience and the size of the sample. As the results of the questionnaire will be used to support the construction of a quality model that is supposed to be used by those interested in architectural activities (our target audience) in the CES field, we expect to generalize them. Therefore, our sample must be a probability sample, according to Kasunic (2005). A probability sample is obtained by randomly selecting members from the population. However, as we do not have a list with every subject of our population, we will select as many subjects that match our criteria established in the previous section.

5.1.4 Design & write questionnaire

In order to ease the application of our questionnaire, we decided to use the online survey tool LimeSurvey \(^1\) installed in a server of our institute. The structure of the final web questionnaire is the following:

1st Page - Questionnaire Introduction

In the first page we begin stating the estimated time required to fulfill the questionnaire (30 minutes). Then, we briefly describe this research project, its goals and the role of the questionnaire within it. Little information about the four parts of the questionnaire is also given. The survey deadlines are presented: 1st reminder (January 24, 2017), 2nd reminder (January 30, 2017), and Deadline (January 31, 2017). Finally, we make it clear that the identity of each respondent will be preserved along with their responses for each question (only the combination of data from each respondent will be published).

2nd Page - Participant Information

To characterize the subjects of our sample and enable data analysis by group, we will ask respondents the following information:

- Age;
- Country of residence;
- Education level (graduate, master, phd, post-doc);
- Current occupation;

\(^1\) <https://www.limesurvey.org/>
5.1. Design of questionnaires

- Application domains and type of systems in which the participant have experience developing/researching CES;
- Years of experience developing/researching CES;
- Classification of previous experience with CES;
- Experience with architectural process (analysis, synthesis and evaluation; and
- Experience/knowledge with architectural methods (e.g., ATAM, DCAR, QAW).

3rd Page - Quality Attributes

In this page, we start introducing what are quality attributes and the point of view to be assumed as a stakeholder or expert. Additionally, we highlight the role of software architecture in satisfying system qualities and give instructions for participants on how to answer the questions for each quality attribute selected in preliminary model.

After section introduction, we present one question for each quality characteristic selected to compose the preliminary model. The name of the characteristic is given along with its definition taken from ISO/IEC 25010 (2011) and the sub-characteristics that were selected to compose them. Finally, for each sub-characteristic selected, we present it within the characteristic question, its definition and a question that asks the participant to evaluate the sub-characteristic regarding its importance. The evaluation is done by selecting a value in a likert-scale ranging from 1 to 5, as previously explained in Section 2.3.5.

4th Page - Trade-offs

In this page, we start explaining what is a trade-off relation between two quality attributes, the need of designers to find a balance between different quality attributes during software development, and the importance of knowing trade-off relations to make design decisions.

Only one question is asked in this page. Participants should identify at least 3 trade-off relations between attributes we included in our preliminary model. To enable participants to do this task, we developed an interactive version of the preliminary model using HTML, CSS and Javascript. This model was embedded into the question and could be used by participants to see the hierarchy of quality characteristics and mouse-over to read their definitions. Additionally, we provided a table with examples of fulfilment. Basically, participants have to write, for each relation identified, one quality characteristic in column A and another in column B. The interpretation is that the quality characteristic in column A negatively impacts the characteristic in column B, an uni-directional relation.

5th page - Final considerations

The last page begins asking participants if they think the quality model will be useful to support software engineering activities to develop CES. The answer is given in a 5-point
likert-scale: 1 - Strongly disagree; 2 - Disagree; 3 - Neither agree nor disagree; 4 - Agree; and 5 - Strongly agree.

Participants that answered “Neither agree nor disagree” (3) or less have to explain why they disagree or are neutral. Those that selected “Agree” (4) or “Strongly agree” (5) were prompted to select phases of the architectural process and architectural methods that could benefit from the model.

After asking for participants to evaluate the usefulness of the model, we asked them to indicate whether the model is missing any relevant quality attribute for CES. If the answer is “No”, no additional question is asked to evaluate model completeness. Otherwise, an interactive model similar to the one developed and embedded in 4th page is presented with all characteristics and sub-characteristics of ISO/IEC 25010 (2011). Sub-characteristics selected and not selected for the preliminary model were highlighted in green and red, respectively. This way, participants could indicate characteristics of the ISO/IEC 25010 (2011) not selected or any other attribute not present in the standard that they might found relevant. In this case, they are asked to give a definition for the attribute, an optional comment, they importance following the DUMOD scale, and trade-off relations (optionally).

Finally, a text field is left for participants so they can provide any additional information they consider useful for the research.

5.1.5 Pilot test questionnaire

In order to find problems before we distribute the questionnaire for our audience, we asked 2 researchers from our research group that are also part of the population to pilot it as if their answers would be included in final results. These testers could found major problems regarding the questions we desire to be answered. Additionally, we asked 2 researchers not related to this research from our laboratory to pilot the test and report us their thoughts. This helped us identify problems regarding the way we present our questionnaire and other minor problems.

After considering the feedback from both groups of testers we finished the design of our questionnaire. It can be found in Appendix B.

5.1.6 Distribute the questionnaire

The questionnaire will be distributed to every subject at January 17th directly via e-mail (system generated link). Eligible participants for the survey will be found analyzing CV of members of laboratories from our institute that work with CES and individuals that have developed their research within the context of CES. As specified before, we will send a reminder one week after the distribution of questionnaires and one day before deadline to increase response rate.
5.2 Data collection

5.1.7 Analyze results and write report

For data analysis, we will employ R-Studio\(^2\) to preprocess raw data exported from LimeSurvey, generate descriptive statistics, perform factor analysis, and plot charts (with package ggplot2\(^3\)) for analysis and publication. LibreOffice Calc\(^4\) will also be used to support data manipulation during analysis. An HTML report is expected to be done using R Markdown\(^5\).

In the next two sections - Data collection and Data analysis - we report the results of the survey. We follow the order of presentation used by Villalba, Fernández-Sanz and Martínez (2010) and included two new sections in Data analysis: Trade-off between quality characteristics (Section 5.3.4) and Final considerations (Section 5.3.5). In the first, we address the concept of trade-offs, which is present in the survey but is not addressed by DUMOD. In the later, we address aspects from DUMOD (which were slightly modified) and tradeoffs.

5.2 Data collection

As expected, data collection happened for two weeks using the LimeSurvey platform hosted in our research group server. At total, we received 28 answers, most from students and researchers from our institute that are familiar with CES development. Table 10 shows the summary for participants’ information.

Most participants aged from 22 to 40 years (26/28) with only two participants with more than 40 years and no one with less than 21 years. This concentration is due to many participants being Ph.D. students (11/28) or recently finished their master’s and Ph.D. degree. With most participants from Brazil (25/28), we also had three international participants, one from Greece, one from Netherlands and another from USA. Most of them are pursuing their Ph.D. degree (12/28), while that 11/28 already have their Ph.D. degree. Thirteen participants are associated with some institution as professor or lecturer, while that 17 consider themselves as Ph.D. students or researchers. Five are currently working as software engineers.

Regarding application domains, the experience of participants is concentrated between the domains automotive (19/50) and robotics (17/50), followed by avionics (5/50) and healthcare (5/50). This result is due to a high response rate from members of a laboratory from our institute that address mobile robotics in the context of autonomous vehicles.

Almost half (13/28) of the participants have between 3 and 5 years of experience with CES, followed by participants with more than 5 years of experience (8/28) and 6 between 1 and 3 years. Only one participant have less than one year of experience. 21 of 28 (21/28) participants have a minimal experience of 3 years developing or researching CES, which is a considerable

\(^{2}\) <https://www.rstudio.com/>

\(^{3}\) <http://ggplot2.org/>

\(^{4}\) <https://www.libreoffice.org/discover/calc/>

\(^{5}\) <http://rmarkdown.rstudio.com/>
Table 10 – Participant information summary

| Ages                  | F. | %   |
|-----------------------|----|-----|
| less than 21 years    | 0  | 0.00%|
| between 22 and 30 years | 15 | 53.57%|
| between 31 and 40 years | 11 | 39.29%|
| between 41 and 50 years | 2  | 7.14%|
| 51 years or more      | 0  | 0.00%|
| Total                 | 28 | 100.00%|

| Experience            | F. | %   |
|-----------------------|----|-----|
| less than 1 year      | 1  | 3.57%|
| between 1 and 3 years | 6  | 21.43%|
| between 3 and 5 years | 13 | 46.43%|
| more than 5 years     | 8  | 28.57%|
| Total                 | 28 | 100.00%|

| Country               | F. | %   |
|-----------------------|----|-----|
| Brazil                | 25 | 89.29%|
| Greece                | 1  | 3.57%|
| Netherlands           | 1  | 3.57%|
| USA                   | 1  | 3.57%|
| Total                 | 28 | 100.00%|

| Education             | F. | %   |
|-----------------------|----|-----|
| Undergraduate student | 0  | 0.00%|
| Bachelor’s degree     | 0  | 0.00%|
| Master’s student      | 1  | 3.57%|
| Master’s degree       | 4  | 14.29%|
| Ph.D. degree          | 12 | 42.86%|
| Ph.D. student         | 11 | 39.29%|
| Total                 | 28 | 100.00%|

| Occupation            | F. | %   |
|-----------------------|----|-----|
| Undergraduate student | 0  | 0.00%|
| Master’s student      | 1  | 2.78%|
| PhD. candidate        | 9  | 25.00%|
| Researcher            | 8  | 22.22%|
| Lecturer              | 3  | 8.33%|
| Professor             | 10 | 27.78%|
| Software Engineer     | 5  | 13.89%|
| Total                 | 36 | 100.00%|

| Domain                | F. | %   |
|-----------------------|----|-----|
| Automotive            | 19 | 38.00%|
| Robotics              | 17 | 34.00%|
| Avionics              | 5  | 10.00%|
| Healthcare            | 5  | 10.00%|
| Railways              | 2  | 4.00%|
| Industrial            | 2  | 4.00%|
| Logistics             | 0  | 0.00%|
| Total                 | 50 | 100.00%|

The experience of participants was mostly concentrated in academic research (25/28), design (14/28), and development (17/28) in the context of CES, totalling 53/75 of the
5.2. Data collection

Nine (9/28) participants have no experience regarding software architecture process. On the other hand, most of the participants with experience in software architecture process are familiar with architectural analysis (18/28), followed by architectural synthesis (10/28) and architectural evaluation (9/28). Only 5 participants (5/28) indicated familiarity with the three architectural activities (analysis, synthesis, and evaluation). Regarding known methods for software architecture analysis, synthesis and evaluation, 15 (15/28) participants indicated no familiarity with any of them. For those that indicated familiarity, the distribution between the methods was the following: SAAM (8/28), ATAM (8/28), CBAM (4/28), DCAR (4/28), QAS (3/28), ADD (2/28), and PALM (1/28). AIW, ARID and MTW were not present in any of the responses provided.

Regarding examples of CES in which participants had their experience on, we found the following systems (by domain):

- Automotive domain: autonomous navigation system, driver distraction protection, obstacle detection system, hybrid braking system, vision systems (environment mapping and perception), driving solutions, vehicle localization.
- Avionics: autonomous navigation, flight control system, mission control system, ground control station system, UAV
- Healthcare: robotic socio-therapy, diagnostics, telemedicine, virtual microscopy, ambient assisted living, pacemaker
- Railways: safety-critical programs
- Industrial: telemetry, data acquisition systems
- Robotics: service robots, autonomous navigation, obstacle detection, multi robot systems (swarm)

Descriptive statistics for time spent (in minutes) by participants are shown in Table 11. Without removing outliers, participants took from 9.71 minutes to 306 minutes to fulfill the survey, with a mean of 45.93 minutes. In general, considering the mean values, participants spent a little more time ($\delta = 1.27$) identifying trade-offs (G3) than evaluating quality attributes (G2) in likert-scale. As expected, participant information (G1) was the group that took less time from participants. Final considerations group (G4) was the group with more time spent by participants, with a mean of 16.51 minutes. One participant in special took 257.02 minutes to fill this group of questions, making its response time to be an outlier in our data. It made a lot of considerations in the question left for criticism and suggestions therefore justifying the time spent. A comparison between time spent in the survey is shown in Figure 21. Notice that the mentioned outlier is not
shown in Final Considerations and Total boxplots, allowing us to see that participants took less time to fulfill final considerations than groups G2 (QAs evaluations) and G3 (trade-offs).

Table 11 – Descriptive statistics for time spent (in minutes) in survey, divided by time spent in each group and total survey time

| Question Group                      | Min  | Q1   | Median | SD   | Mean  | Q3   | Max   |
|-------------------------------------|------|------|--------|------|-------|------|-------|
| Participant information (G1)        | 1.89 | 2.99 | 5.11   | 3.08 | 5.49  | 6.91 | 11.97 |
| Quality attributes (G2)             | 3.08 | 5.97 | 9.04   | 8.05 | 11.33 | 13.28| 33.71 |
| Trade-offs (G3)                     | 2.81 | 6.58 | 9.44   | 9.31 | 12.60 | 16.24| 39.76 |
| Final considerations (G4)           | 1.37 | 3.14 | 4.89   | 47.85| 16.51 | 9.67 | 257.02|
| Total                               | 9.71 | 25.58| 34.42  | 53.44| 45.93 | 50.80| 306.73|

Figure 21 – Boxplot of time spend to complete each survey question group and total time. Black dots are outliers (two are omitted for better visualization).

5.3 Data analysis

5.3.1 Previous analysis (preliminary analysis)

In this section, we cover the descriptive statistics for data gathered from quality attribute evaluation question group (G2). Table 12 shows basic statistics for each QA addressed in the
survey.

Table 12 – Descriptive statistics for QAs: minimum value, 1st quartile, median, standard deviation (σ), coefficient of variation (cv), mean, mode, 3rd quartile, and max value

| Quality Attribute             | Min  | Q1   | Median | σ   | cv   | Mean  | Mode | Q3   | Max  |
|-------------------------------|------|------|--------|-----|------|-------|------|------|------|
| Availability                  | 3.00 | 3.00 | 4.00   | 0.79| 0.20 | 3.96  | 4.00 | 5.00 | 5.00 |
| Fault tolerance               | 1.00 | 3.75 | 4.50   | 1.08| 0.26 | 4.14  | 5.00 | 5.00 | 5.00 |
| Maturity                      | 2.00 | 4.00 | 4.00   | 0.90| 0.22 | 4.07  | 4.00 | 5.00 | 5.00 |
| Recoverability                | 1.00 | 3.75 | 4.00   | 1.07| 0.27 | 3.96  | 4.00 | 5.00 | 5.00 |
| Capacity                      | 1.00 | 3.00 | 4.00   | 1.12| 0.30 | 3.71  | 4.00 | 5.00 | 5.00 |
| Resource utilization          | 1.00 | 3.00 | 4.00   | 1.04| 0.28 | 3.75  | 4.00 | 4.25 | 5.00 |
| Time behaviour                | 2.00 | 4.00 | 4.00   | 0.81| 0.19 | 4.29  | 5.00 | 5.00 | 5.00 |
| Learnability                  | 1.00 | 3.00 | 4.00   | 1.25| 0.34 | 3.64  | 5.00 | 5.00 | 5.00 |
| Operability                   | 2.00 | 3.00 | 3.00   | 1.00| 0.29 | 3.50  | 3.00 | 4.00 | 5.00 |
| User error protection         | 2.00 | 3.00 | 4.00   | 1.02| 0.26 | 3.93  | 5.00 | 5.00 | 5.00 |
| Analyzability                 | 2.00 | 3.00 | 3.00   | 0.69| 0.20 | 3.39  | 3.00 | 4.00 | 5.00 |
| Modifiability                 | 2.00 | 3.00 | 4.00   | 0.93| 0.24 | 3.86  | 4.00 | 4.25 | 5.00 |
| Modularity                    | 2.00 | 3.00 | 4.00   | 0.86| 0.22 | 3.93  | 4.00 | 5.00 | 5.00 |
| Testability                   | 2.00 | 3.00 | 4.00   | 0.98| 0.26 | 3.71  | 4.00 | 4.25 | 5.00 |
| Authenticity                  | 2.00 | 3.00 | 3.00   | 0.96| 0.28 | 3.39  | 3.00 | 4.25 | 5.00 |
| Confidentiality               | 2.00 | 3.00 | 4.00   | 1.14| 0.30 | 3.75  | 5.00 | 5.00 | 5.00 |
| Integrity                     | 2.00 | 3.00 | 4.50   | 1.14| 0.28 | 4.04  | 5.00 | 5.00 | 5.00 |
| Non-repudiation               | 2.00 | 3.00 | 4.00   | 0.87| 0.24 | 3.64  | 3.00 | 4.00 | 5.00 |
| Co-existence                  | 1.00 | 3.00 | 4.00   | 1.10| 0.30 | 3.61  | 3.00 | 4.25 | 5.00 |
| Interoperability              | 1.00 | 3.00 | 4.00   | 1.02| 0.27 | 3.82  | 4.00 | 5.00 | 5.00 |
| Adaptability                  | 1.00 | 2.75 | 3.00   | 1.12| 0.34 | 3.32  | 3.00 | 4.00 | 5.00 |
| Trust                         | 3.00 | 4.00 | 4.50   | 0.77| 0.18 | 4.32  | 5.00 | 5.00 | 5.00 |
| Economic risk                 | 2.00 | 3.00 | 4.00   | 1.08| 0.29 | 3.71  | 5.00 | 5.00 | 5.00 |
| Environmental risk            | 2.00 | 3.00 | 4.00   | 1.03| 0.26 | 3.89  | 5.00 | 5.00 | 5.00 |
| Health risk                   | 2.00 | 3.75 | 5.00   | 1.02| 0.24 | 4.32  | 5.00 | 5.00 | 5.00 |

Regarding minimum and maximum values, every attribute achieved at least one score of 5 (max possible - always), while that 8 achieved the minimum value of 1 (never) at least once. 15 attributes achieved at least one minimum score of 2 (rarely) and only two attributes achieved at least one minimum score of 3 (possibly): Trust and Health and Safety risk mitigation. Both also scored best with 4,32 mean values each. Adaptability achieved the lowest mean: 3,32. Figure 22 shows the barplot of means achieved by each QA, grouped by colors that are representing the associated quality characteristic.

Regarding the coefficient of variation, the lower and higher value was 0,18 (Trust) and 0,34 (Learnability), respectively. The boxplots of Figure 24 show high concentration of values for each QA between the scores 3 and 5, with only a few outliers. This concentration can also be seen in Table 13.

Considering the totals, almost half of the scores given by participants for each QA were 5 (44,27%), followed by score 4 (31,35%), 3 (18,28%), 2 (5,51%), and 1 (0,59%).
Chapter 5. Quality Model Evaluation

The correlation matrix computed for quality attributes is shown in Figure 25. Correlation coefficients with no significance \((\alpha = 0.05)\) are not shown in this matrix (empty squares). We identified a significant number of correlations between QAs with a coefficient higher than 0.30. The determinant of this matrix is 3.375766e-14, which is an indicator that the variables are highly correlated between them.

### 5.3.2 Reliability analysis

Reliability analysis was performed within the following characteristics: Reliability, Performance efficiency, Usability, Maintainability, Security, and Freedom from Risk. Compatibility, Portability and Satisfaction characteristics were not considered for reliability analysis as they are composed by one or two quality attributes. In general, most characteristics achieved a considerable level of internal consistence considering the Cronbach’s alpha, except Reliability, which was classified with Poor/Unacceptable consistence. Security and Freedom from Risk achieved Good internal consistence while that Performance efficiency, Usability, and Maintainability achieved an Acceptable level.

For Reliability characteristic, dropping Recoverability will increase internal consistency from Unacceptable/Poor (0.49/0.52) to Questionable (0.65). For Performance efficiency, dropping
### 5.3. Data analysis

| Quality Attributes |
|--------------------|
| Availability       |
| Fault tolerance    |
| Maturity           |
| Recoverability     |
| Capacity           |
| Resource utilization|
| Time behaviour     |
| Learnability       |
| Usability          |
| Security           |
| Authenticity       |
| Confidentiality    |
| Interoperability   |
| Co-existence       |
| Non repudiation    |
| Integrity          |
| Modifiability      |
| Testability        |
| Analyzability      |
| User error protection|
| Operability        |
| Modularity         |

**Figure 23 – Boxplot of each QA regarding score achieved in likert-scale**

Time behavior will improve consistency from Acceptable (0.76/0.75) to Good (0.84/0.85). From Usability, dropping User error protection slightly improve internal consistency by a few points, from Acceptable (0.77/0.78) to Good (0.80/0.81). This is also the case of Analyzability within Security characteristic. Dropping that QA improves internal consistency by a few points, from Acceptable (0.78) to Good (0.80). Finally, for Security and Freedom from Risk characteristics, dropping Authenticity and Health risk mitigation barely improves Cronbach's alpha, with no gain to internal consistency level. If dropped, authenticity improves alpha of Security by 0.03 (from 0.85 to 0.88). In the case of Health risk, this gain is even lower: 0.01 (from 0.81 to 0.82). None of the characteristics analyzed presented a Cronbach’s alpha higher than 0.85, therefore we had no characteristic classified with Excellent internal consistency.

Regarding item-total correlations, no QAs from Reliability characteristic achieved the recommended value of 0.7, with Recoverability achieving the lowest score (0.2) within all characteristics. Maintainability, in second place, had just one QA that achieved the minimum score: Modularity. The others QAs were very near that minimum (Analyzeability: 0.52; Modifiability: 0.69; and Testability: 0.64). For the remaining quality characteristics, Performance efficiency, usability, and security, only the QAs that increase Cronbach’s alpha within their group did not achieved the minimum recommended score.
Figure 24 – Boxplot of each QA regarding score achieved in likert-scale

Table 14 shows information from executing reliability analysis.

5.3.3 Predictive validation (exploratory factorial analysis)

To measure the applicability of factorial analysis, Kaiser-Meyer-Olkin (KMO) index and Barlett’s test of sphericity were checked. The score achieved for KMO index was 0.37 (unacceptable). Values higher than 0.5 are suggested as indicator that factorial analysis outcomes are relevant. Otherwise, the Barlett’s test yield a chi-square of 553.1823, with a p-value of 2.905872e-17.

With these values, we decided not to proceed with exploratory factorial analysis.
Table 13 – Relative frequency of values for each QA in likert-scale

| Quality Attribute    | 1       | 2       | 3       | 4       | 5       |
|----------------------|---------|---------|---------|---------|---------|
| Availability         | 0,00%   | 0,00%   | 32,14%  | 39,29%  | 28,57%  |
| Fault tolerance      | 3,57%   | 3,57%   | 17,86%  | 25,00%  | 50,00%  |
| Maturity             | 0,00%   | 7,14%   | 14,29%  | 42,86%  | 35,71%  |
| Recoverability       | 3,57%   | 7,14%   | 14,29%  | 39,29%  | 35,71%  |
| Capacity             | 3,57%   | 10,71%  | 25,00%  | 32,14%  | 28,57%  |
| Resource utilization | 3,57%   | 7,14%   | 25,00%  | 39,29%  | 25,00%  |
| Time behaviour       | 0,00%   | 3,57%   | 10,71%  | 39,29%  | 46,43%  |
| Learnability         | 3,57%   | 17,86%  | 25,00%  | 17,86%  | 35,71%  |
| Operability          | 0,00%   | 14,29%  | 42,86%  | 21,43%  | 21,43%  |
| User error protection| 0,00%   | 10,71%  | 21,43%  | 32,14%  | 35,71%  |
| Analyzability        | 0,00%   | 7,14%   | 50,00%  | 39,29%  | 3,57%   |
| Modifiability        | 0,00%   | 10,71%  | 17,86%  | 46,43%  | 25,00%  |
| Modularity           | 0,00%   | 3,57%   | 28,57%  | 39,29%  | 28,57%  |
| Testability          | 0,00%   | 10,71%  | 32,14%  | 32,14%  | 25,00%  |
| Authenticity         | 0,00%   | 17,86%  | 39,29%  | 28,57%  | 14,29%  |
| Confidentiality      | 0,00%   | 17,86%  | 25,00%  | 21,43%  | 35,71%  |
| Integrity            | 0,00%   | 14,29%  | 17,86%  | 17,86%  | 50,00%  |
| Non-repudiation      | 0,00%   | 7,14%   | 39,29%  | 35,71%  | 17,86%  |
| Co-existence         | 3,57%   | 10,71%  | 32,14%  | 28,57%  | 25,00%  |
| Interoperability     | 3,57%   | 3,57%   | 28,57%  | 35,71%  | 28,57%  |
| Adaptability         | 3,57%   | 21,43%  | 32,14%  | 25,00%  | 17,86%  |
| Trust                | 0,00%   | 0,00%   | 17,86%  | 32,14%  | 50,00%  |
| Economic risk        | 0,00%   | 14,29%  | 32,14%  | 21,43%  | 32,14%  |
| Environmental risk   | 0,00%   | 10,71%  | 25,00%  | 28,57%  | 35,71%  |
| Health risk          | 0,00%   | 7,14%   | 17,86%  | 10,71%  | 64,29%  |
| Total                | 0,59%   | 5,51%   | 18,28%  | 31,35%  | 44,27%  |

5.3.4 Trade-off between quality characteristics

Regarding trade-off relations between attributes, we asked each participant to identify at least three relations. We gathered a total of 115 relations submitted. 11 participants (11/28) submitted the minimum of 3 relations, 3 participants (3/28) submitted 5 relations, 8 participants (8/28) submitted 6, 2 participants (2/28) submitted 7, 2 participants (2/28) submitted 10, 1 submitted (1/28) 12, and 1 (1/28) submitted 14.

When counting the frequency of pair of relations submitted by participants, we had the following results: the pair (Interoperability, Confidentiality) was submitted by 7 participants and was the greater frequency we found; The pairs (Co-existence, Integrity) and (Interoperability, Integrity) had both a frequency of 4; following, with a frequency of 3, were the pairs (Fault tolerance, Time behaviour), (Modularity, Integrity), (User error protection, Time behaviour), (Adaptability, Time behaviour), (Interoperability, Testability), (Co-existence, Confidentiality), (Adaptability Resource utilization), and (Adaptability, Testability). With a frequency of 2, we found 14 pairs, (Co-existence, Time behaviour), (Capacity, Resource utilization), (Time behaviour, Maturity),
Figure 25 – Correlation matrix for QAs scores. It is showing only correlation coefficients between QAs that achieved a significance level of 0.05 (p-value)

(Availability, Confidentiality), (Fault tolerance, Resource utilization), (Co-existence, Capacity), (Maturity, Integrity), (Authenticity, Time behaviour), (Co-existence, Testability), (Economic Risk Mitigation, Environmental Risk Mitigation), (Fault tolerance, Adaptability), (Modularity, Time behaviour), (Adaptability, Economic Risk Mitigation), and (Adaptability, Capacity). Finally, 90 relations were submitted only one time among all participants: (Trust, Availability), (Resource utilization, Fault tolerance), (Availability, Capacity), (Health and Safety Risk Mitigation, Operability), (Availability, Testability), (Authenticity, Time Behaviour), (Operability, Adaptability), (Fault tolerance, Learnability), (Time behaviour, Modularity), (Availability, Trust), (Learnability, Modifiability), (Non-repudiation, Operability), (Fault Tolerance, Time Behaviour), (Capacity, Modifiability), (Time behaviour, Non-repudiation), (Interoperability, Fault Tolerance), (Integrity, Modifiability), (Trust, Adaptability), (Fault tolerance, Modifiability), (Interoperability, Fault
Table 14 – Computed values from reliability analysis.

| Quality Attribute | Raw alpha if drop. | Std. alpha if drop. | Item-total corr. | Group raw alpha | Group std. alpha |
|-------------------|--------------------|---------------------|------------------|----------------|-----------------|
| **Reliability**   |                    |                     |                  |                |                 |
| Availability      | 0.21               | 0.22                | 0.68             | 0.49           | 0.52            |
| Fault tolerance   | 0.22               | 0.28                | 0.64             |                |                 |
| Maturity          | 0.47               | 0.51                | 0.43             |                |                 |
| Recoverability    | 0.65               | 0.65                | 0.20             |                |                 |
| **Performance Efficiency** |    |                     |                  |                |                 |
| Capacity          | 0.64               | 0.65                | 0.75             | 0.76           | 0.75            |
| Resource utilization | 0.44             | 0.46                | 0.87             |                |                 |
| Time behaviour    | 0.84               | 0.85                | 0.49             |                |                 |
| **Usability**     |                    |                     |                  |                |                 |
| Learnability      | 0.71               | 0.71                | 0.72             | 0.77           | 0.78            |
| Operability       | 0.55               | 0.56                | 0.84             |                |                 |
| User error protection | 0.80              | 0.81                | 0.58             |                |                 |
| **Maintainability** |                    |                     |                  |                |                 |
| Analyzability     | 0.80               | 0.80                | 0.52             | 0.78           | 0.78            |
| Modifiability     | 0.71               | 0.71                | 0.69             |                |                 |
| Modularity        | 0.60               | 0.61                | 0.90             |                |                 |
| Testability       | 0.76               | 0.77                | 0.64             |                |                 |
| **Security**      |                    |                     |                  |                |                 |
| Authenticity      | 0.88               | 0.88                | 0.56             | 0.85           | 0.85            |
| Confidentiality   | 0.76               | 0.77                | 0.89             |                |                 |
| Integrity         | 0.79               | 0.79                | 0.84             |                |                 |
| Non repudiation   | 0.81               | 0.80                | 0.77             |                |                 |
| **Freedom from Risk** |                |                     |                  |                |                 |
| Economic risk     | 0.80               | 0.80                | 0.70             | 0.81           | 0.81            |
| Environmental risk | 0.56               | 0.56                | 0.90             |                |                 |
| Health risk       | 0.82               | 0.82                | 0.67             |                |                 |

tolerance), (Resource utilization, Capacity), (Maturity, Modifiability), (Trust, Co-existence), (Integrity, Modularity), (Confidentiality, Time Behaviour), (Trust, Fault tolerance), (Time behaviour, Learnability), (Interoperability, Operability), (Learnability, Modularity), (Adaptability, Modifiability), (Modifiability, Integrity), (Availability, User error protection), (Integrity, Co-existence), (Adaptability, Fault Tolerance), (Capacity, Economic Risk Mitigation), (Adaptability, Fault tolerance), (Learnability, Operability), (Trust, Integrity), (Integrity, Time Behaviour), (Economic Risk Mitigation, Learnability), (Learnability, Testability), (Modularity, Maturity), (Adaptability, Authenticity), (Co-existence, Resource utilization), (Testability, Time behaviour), (Non-repudiation, Interoperability), (Health and Safety Risk Mitigation, Modifiability), (User error protection, Learnability), (Capacity, Environmental Risk Mitigation), (Recoverability, Confidentiality), (Non-repudiation, Time Behaviour), (Health and Safety Risk Mitigation, Learnability), (Resource utilization, Interoperability), (Modifiability, Maturity), (Adaptability, Analyzability), (Capacity, Trust), (Modularity, Resource utilization), (Health and Safety Risk Mitigation, Modularity), (Capacity, Health and Safety Risk Mitigation), (Confidentiality, Interoperability), (Environmental Risk Mitigation, Learnability), (Co-existence, Trust), (Maturity, Trust), (Confidentiality, Time behaviour), (Testability, Modularity), (Recoverability, Resource utilization), (Health and Safety Risk Mitigation, Adaptability), (Time behaviour, Trust), (Integrity, Analyzability), (Authenticity, Non-repudiation), (Availability, Learnability), (User error protection, Trust), (Co-existence, Fault tolerance), (Economic Risk Mitigation, Co-existence), (User error protection, Recoverability), (Recoverability, Time behaviour), (User error protection, Co-existence), (Modifiability, Adapt-
ability), (Adaptability, Time Behaviour), (Fault tolerance, Operability), (User error protection, Interoperability), (Confidentiality, Trust), (Non-repudiation, Time behaviour), (Learnability, Confidentiality), (Authenticity, Trust), (Confidentiality, Non-repudiation), (Testability, Maturity), (Co-existence, Authenticity), (Trust, Resource utilization), and (Co-existence, Non-repudiation).

Checking the relations submitted, we identified that 26 relations had at least one inverted version of it. It means that we can consider them bi-directional trade-off as we asked participants to identify only uni-directional relations. The list of bi-directional relations is given below:

- Interoperability ⇔ Confidentiality
- Co-existence ⇔ Integrity
- Modularity ⇔ Integrity
- Capacity ⇔ Resource utilization
- Fault tolerance ⇔ Resource utilization
- Fault tolerance ⇔ Adaptability
- Modularity ⇔ Time behaviour
- Trust ⇔ Availability
- Resource utilization ⇔ Fault tolerance
- Time behaviour ⇔ Modularity
- Availability ⇔ Trust
- Time behaviour ⇔ Non-repudiation
- Integrity ⇔ Modifiability
- Resource utilization ⇔ Capacity
- Maturity ⇔ Modifiability
- Trust ⇔ Co-existence
- Integrity ⇔ Modularity
- Adaptability ⇔ Modifiability
- Modifiability ⇔ Integrity
- Integrity ⇔ Co-existence
- Adaptability ⇔ Fault tolerance
5.3. Data analysis

- Modifiability ⇔ Maturity
- Confidentiality ⇔ Interoperability
- Co-existence ⇔ Trust
- Modifiability ⇔ Adaptability
- Non-repudiation ⇔ Time behaviour

For a better view on the relations submitted by participants, we draw two illustrations. Figure 26 shows the relations between sub-characteristics submitted by at least two participants. Figure 27 shows the same relations from Figure 26 organized by characteristics instead of sub-characteristics. Analyzing both figures, we can see that the most impacted characteristics are Performance Efficiency and Security. Performance Efficiency was not indicated to be affected only by Freedom from Risk, being impacted in 24 relations with other sub-characteristics. On the other hand, Compatibility was considered the most impactful on other characteristics, with 18 relations with Security, 5 with Maintainability and 4 with Performance Efficiency.

5.3.5 Final considerations

In final considerations page, we asked participants to give us feedback regarding the usefulness of the quality model, a rationale for usefulness evaluation (reason in case of score 3 or less, or selection of architectural activities if score is 4 or 5), their opinion on model completeness, an optional question to let participants submit quality attributes not considered in the model and additional information regarding suggested attributes, and finally, a text area for any additional information participants consider relevant to this research.

Concerning model usefulness, most participants (71%) agree that the model can be useful to support the development of CES, while that five participants (18%) were neutral, two strongly agree (7%) and one disagree (4%). No participant answered strongly disagree. It means that 22 participants had to select architectural phases and activities that they think can be supported by QM4CES and 6 had to give a rationale regarding their answer.

Summarizing the answers of 22 participants that agree on usefulness of the model, architectural evaluation was the process most selected (19/22), followed by architectural analysis (15/22), and architectural synthesis (12/22). Considering architectural methods from literature, ATAM and SAAM were the most selected methods with 11 out of 22 participant’s answers. CBAM had 8 selections, followed by ADD (7), QAW (5), DCAR (4), PALM, ARID, AIW (3 each), and MTW (1).

The answers from the participants that disagree or are neutral regarding the model were giving in text format. In summary three of the six participants commented that the information provided in the survey was not sufficient to allow a coherent evaluation. One participant pointed
that the model is too much general and theoretical, therefore making it hard to support CES development. Another participant criticized the way software engineering is addressing the development of CES as focus on hardware is also required in her/his point-of-view. Finally, one participant was not sure about the answer he/she given regarding model usefulness.

Concerning model completeness, 16 participants (57%) pointed that the model was not missing any quality attribute, 12 (43%) pointed the contrary. For these 12 participants, a set of new questions were given so they should have pointed additional quality attributes, evaluate them using the DUMOD scale and, optionally, indicate trade-off relations of these attributes.

In total, participants suggested 26 attributes. Two high-level characteristics were suggested (Usability and Functional Suitability) although we asked participants only to suggest sub-characteristics. Maybe we could be more clear about which characteristics should be suggested or use input validation to solve this problem. Resource utilization and confidentiality were suggested although both were present in the model. The later was suggested because of an error in the interactive model, which was showing that confidentiality was not selected. Regardless of
5.3. Data analysis

Figure 27 – Relations submitted by participants. The number labelling the lines represents the number of times the relation was submitted. The color of the lines and their thickness is based on their frequency: green (2 times), blue (3-7 times), golden (8 times), and red (18 times).

These problems, 11 characteristics from ISO/IEC 25010 (2011) itself were suggested: Functional Correctness was suggested 6 times, followed by Reusability (5 times), Functional Completeness and Functional and Replaceability (2 times each), and Accessibility, Comfort, Effectiveness, Efficiency, Flexibility, Installability, and Pleasure (1 time each). Summary statistics for new characteristics is shown in Table 15.

| Quality Attribute          | Mean | Median | Mode |
|----------------------------|------|--------|------|
| Functional Correctness     | 4.33 | 4.5    | 5    |
| Reusability                | 4    | 4      | 3    |
| Functional Completeness    | 5    | 5      | 5    |
| Replaceability             | 5    | 5      | 5    |
5.3.6 Discussion of results

Regarding the acceptability of quality characteristics included in the preliminary model, we understood that most participants agree that all of them should be, in fact, in the model. This is supported by the high mean values and low coefficient of variation achieved for sub-characteristics.

As expected, Health and Safety risk mitigation (a.k.a. safety) had the greatest mean, tied with Trust. Surprisingly, Trust had the lowest standard deviation and coefficient of variation. We did not expect such a result since we did not find any study addressing this characteristic in the SM. Health and Safety risk mitigation had the most frequency of answers "always". This is an indicator of consensus of participants regarding this sub-characteristic. Time behaviour (a.k.a performance) achieved the third highest mean. This is one of the most important characteristic for CES, considering the number of studies addressing it in SM, and its relation to real-time computing.

In general, sub-characteristics of Reliability were on top of means. This result was expected to support the large number of studies in SM addressing Reliability and its sub-characteristics.

Resource utilization, although addressed in many studies of the SM, was not well positioned within the means. Perhaps the participants were not able to link this sub-characteristic to the necessity of minimizing power consumption of CES. Moreover, most of our participants were academic researchers. We believe that energy consumption was not a problem in this context.

Regarding internal consistency, except for Reliability, every other characteristic achieved a high value, indicating high internal consistency. In a few cases, dropping sub-characteristics could increase Cronbach’s alpha, but it would not have made sense removing, for example, Recoverability or Time behaviour.

Unfortunately, our data was not suitable for factorial analysis, according to the KMO index. In fact, factor analysis are normally performed by researchers on samples of at least 50 observations, and preferably the sample size should be of 100 observations or more (HAIR, 2010).
CES are of utmost important for the current society. They are already present in many application domains and will be increasingly employed in areas that have not been yet explored. Although this might be positive for many aspects of the society, CES developers should be aware of their importance and employ suitable approaches to ensure that no harm is done to living beings or the environment.

In this work, we established a quality model for CES, which contains a subset of characteristics selected (based on evidence found in a SM) from ISO/IEC 25010 (2011). The established model, QM4CES, was further presented to experts (mostly researchers) in the field for an evaluation regarding model usefulness and completeness. The majority of participants agreed that the model is useful to support the development of CES, and suggested a few list of new characteristics from the own ISO/IEC 25010 (2011) to be included in the final version of the model. Moreover, participants provided the most challenging tradeoffs faced during CES development.

We concluded that quality is a very complex concept that can be used for almost anything within the society. From natural resources to manufactured products, we can find quality concepts related to them. Knowing these concepts is important to support decision making of every kind. For software it is not different. As presented in Chapter 2, many authors identified factors that describe the overall quality of software products, be it from the point-of-view of the developer or the end user. Such factors, also called as quality attributes or quality characteristics, developed their own community within the software engineering community. Many approaches for achieving them can be found in the literature, making it hard to have a broad view of every community. As consequence, we found a lot of different QA in the SM, addressed by different ways. Many studies did not provided a formal definition for QA addressed, making it even hard to figure out what exactly they were talking about. Performance, for example, is decomposed by ISO/IEC 25010 (2011) into three attributes with different meanings and metrics associated to them. The mapping procedure performed to map different attributes into ISO/IEC 25010 (2011)
shows a significant value to understand the QA addressed in CES development.

Another relevant aspect regarding quality is: who best knows what defines the quality of a CES, for instance? The answer is not simple, but we find out that the literature can be explored through systematic reviews/mappings, being a very relevant source of QM, QA, and quality metrics. Standards are also very useful sources, however, most of them are not free available for use. Although literature can be useful, the opinion of domain experts from industry might be more valuable. In general, research tend to focus on some QA while that industries have to face many QA requirements simultaneously to project constraints and time to market.

For future work, we intend to elaborate more on several concepts that can be related to QA within the model to increase its usefulness for different communities. For instance, to elaborate and relate quality scenarios to QA of QM4CES can be very valuable for software architecture and requirements communities, while that definition of metrics can benefit product evaluators or support quality management.
AA1CAR. How to Diagnose a Controller Area Network (CAN). 2014. <http://www.aa1car.com/library/can_systems.htm>. Citation on page 33.

AGUIAR, A.; CASAGRANDE, T. D.; HESSEL, F.; ALEGRE, P. Hellfire: A Design Framework for Critical Embedded Systems’ Applications. In: 11th International Symposium on Quality Electronic Design (ISQED). San Jose, CA, USA: [s.n.], 2010. p. 730–737. Citations on pages 21, 26, and 79.

AHRENS, D.; FREY, A.; PFEIFFER, A.; BERTRAM, T. Objective evaluation of software architectures in driver assistance systems. Computer Science - Research and Development, v. 28, n. 1, p. 23–43, 2011. Citations on pages 22, 61, and 62.

_____ . Objective evaluation of software architectures in driver assistance systems. Computer Science-Research and Development, Springer, v. 28, n. 1, p. 23–43, 2013. Citations on pages 66, 87, 88, 90, 91, and 92.

AL-DAAJEH, S. H.; AL-QUTAISH, R. E.; AL-QIREM, F. Engineering dependability to embedded systems software via tactics. Citeseer, 2011. Citations on pages 66, 87, 88, 90, and 92.

ANDALAM, S.; ROOP, P.; GIRAULT, A.; TRAULSEN, C. A Predictable Framework for Safety-Critical Embedded Systems. IEEE Transactions on Computers, v. 63, p. 1600–1612, 2013. Citation on page 21.

ANDREOU, A. S.; TZIAKOURIS, M. A quality framework for developing and evaluating original software components. Information and Software Technology, v. 49, n. 2, p. 122–141, 2007. Citation on page 56.

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: IEEE. 2011 18th Asia-Pacific Software Engineering Conference. [S.l.], 2011. p. 211–218. Citation on page 65.

ARMOUSH, A.; SALEWSKI, F.; KOWALEWSKI, S. Effective Pattern Representation for Safety Critical Embedded Systems. In: International Conference on Computer Science and Software Engineering. Wuhan, Hubei: [s.n.], 2008. p. 91–97. Citation on page 21.

AVIŽIENIS, A.; LAPRIE, J. C.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing, v. 1, n. 1, p. 11–33, Jan 2004. Citation on page 72.

AXELSSON, J. Evolutionary architecting of embedded automotive product lines: An industrial case study. In: IEEE. Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on. [S.l.], 2009. p. 101–110. Citations on pages 66, 87, 88, 90, 91, and 92.
BACHMANN, F.; BASS, L.; KLEIN, M.; SHELTON, C. Designing software architectures to achieve quality attribute requirements. IEEE Software, v. 152, n. 4, p. 153–165, 2005. Citation on page 48.

BANSIYA, J.; DAVIS, C. A hierarchical model for object-oriented design quality assessment. IEEE Transactions on Software Engineering, v. 28, n. 1, p. 4–17, 2002. Citation on page 55.

BARBACCI, M. R.; ELLISON, R.; LATTANZE, A. J.; STAFFORD, J. A.; WEINSTOCK, C. B.; WOOD, W. G. Quality Attribute Workshops (QAWs), Third Edition. [S.l.], 2003. Citation on page 40.

BARBACCI, M. R.; KLEIN, M. H.; LONGSTAFF, T. A.; WEINSTOCK, C. B.; OTHERS. Quality Attributes. [S.l.], 1995. Citations on pages 15, 49, and 72.

BARNEY, S.; PETERSEN, K.; SVAHNBERG, M.; AURUM, A.; BARNEY, H. Software quality trade-offs: A systematic map. Information and Software Technology, Elsevier B.V., v. 54, n. 7, p. 651–662, 2012. Citation on page 51.

BASILI, V. R. Software modeling and measurement: the Goal/Question/Metric paradigm. [S.l.], 1992. 24 p. Citation on page 98.

BASS, L.; CLEMENTS, P.; KAZMAN, R. Software Architecture in Practice. 3rd. ed. [S.l.]: Addison-Wesley, 2012. Citations on pages 17, 22, 36, 37, 38, 39, 48, 49, and 52.

BASS, L.; JOHN, B. E. Supporting usability through software architecture. Computer, v. 34, n. 10, p. 113–115, 2001. Citation on page 89.

BATE, I. Systematic approaches to understanding and evaluating design trade-offs. Journal of Systems and Software, v. 81, n. 8, p. 1253–1271, 2008. Citations on pages 26, 49, and 79.

BEHKAMAL, B.; KAHANI, M.; AKBARI, M. K. Customizing ISO 9126 quality model for evaluation of B2B applications. Information and Software Technology, v. 51, n. 3, p. 599–609, 2009. Citations on pages 56 and 77.

BENYAHIA, I.; POTVIN, J.-Y.; XU, Y. A complex applications framework supporting tolerant dynamic vehicle dispatching. In: IEEE. IEEE International Conference on Computer Systems and Applications, 2006. [S.l.], 2006. p. 875–881. Citations on pages 65 and 88.

BOEHM, B. W.; BROWN, J. R.; LIPOW, M. Quantitative Evaluation of Software Quality. In: 2nd International Conference on Software Engineering (ICSE 1976). San Francisco, California, USA: [s.n.], 1976. p. 592–605. Citations on pages 21, 22, 48, 55, 56, 67, and 77.

BOSCH, J. Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach. [S.l.]: Addison-Wesley, 2000. Citation on page 37.

CALERO, C.; CACHERO, C.; CÓRDOBA, J.; MORAGA, M. Pqm vs. bpqm: Studying the tailoring of a general quality model to a specific domain. In: Advances in Conceptual Modeling – Foundations and Applications. [S.l.]: Springer Berlin Heidelberg, 2007. p. 192–201. Citation on page 56.

CARVALHO, F.; MEIRA, S. Towards an Embedded Software Component Quality Verification Framework. In: 14th International Conference on Engineering of Complex Computer Systems. Potsdam, DE: [s.n.], 2009. p. 248–257. Citations on pages 22, 61, and 63.
CENELEC-50128. Railway Applications: Software for Railway Control and Protection Systems. 2000. Citation on page 35.

CENELEC-50129. Railway Applications: Software for Railway Control and Protection Systems. 1999. Citation on page 35.

CHAN, C. A Treatise on Crash Sensing for Automotive Air Bag Systems. Mechatronics, IEEE/ASME Transactions on, v. 7, n. 2, p. 220–234, 2002. Citation on page 34.

CHAO, W.; JIANMIN, D.; YONGCHUAN, Y. A hardware in loop test system for pneumatic anti-lock brake system. In: 2nd International Conference on Measuring Technology and Mechatronics Automation (ICMTMA 2010). Changsha City, CH: IEEE Computer Society Press, 2010. p. 105–108. Citation on page 34.

CHEUNG, L.; ROSHANDEL, R.; MEDVIDOVIC, N.; GOLUBCHIK, L. Early prediction of software component reliability. In: 2008 ACM/IEEE 30th International Conference on Software Engineering. [S.l.: s.n.], 2008. p. 111–120. Citation on page 71.

CHIPRIANOV, V.; FALKNER, K.; SZABO, C.; PUDDY, G. Architectural support for model-driven performance prediction of distributed real-time embedded systems of systems. In: SPRINGER. European Conference on Software Architecture. [S.l.], 2014. p. 357–364. Citation on page 65.

CLEMENTS, P.; KAZMAN, R.; KLEIN, M. Evaluating Software Architectures: Methods and Case Studies. [S.l.]: Addison-Wesley, 2002. Citation on page 51.

DEISSENBOECK, F.; JUERGENS, E.; LOCHMANN, K.; WAGNER, S. Software quality models: Purposes, usage scenarios and requirements. In: Workshop on Software Quality (WOSQ 2009). Vancouver, BC: [s.n.], 2009. p. 9–14. Citations on pages 21, 54, and 55.

DIPOPPA, G.; D’ALESSANDRO, G.; SEMPRINI, R.; TRONCI, E. Integrating automatic verification of safety requirements in railway interlocking system design. In: Proceedings Sixth IEEE International Symposium on High Assurance Systems Engineering. Special Topic: Impact of Networking. [S.l.: s.n.], 2001. Citation on page 35.

DOBRICA, L.; NIEMELÄ, E. An Approach to Reference Architecture Design for Different Domains of Embedded Systems. In: International Conference on Software Engineering Research & Practice (SERP 2008). Las Vegas, Nevada, USA: [s.n.], 2008. p. 287–293. Citations on pages 25 and 27.

DOD. Systems engineering guide for system of systems. [S.l.], 2008. Citation on page 91.

DROMEY, R. G. A Model for Software Product Quality. IEEE Transactions on Software Engineering, v. 21, n. 2, p. 146–162, 1995. Citations on pages 22, 55, 56, and 77.

DU, S.; YU, D.; HU, Y.; HUANG, Y. A component model and layered system architecture for reconfigurable cnc systems. In: IEEE. Information and Automation, 2009. ICIA’09. International Conference on. [S.l.], 2009. p. 922–927. Citations on pages 65, 88, and 92.

DURISIC, D.; NILSSON, M.; STARON, M.; HANSSON, J. Measuring the impact of changes to the complexity and coupling properties of automotive software systems. Journal of Systems and Software, Elsevier, v. 86, n. 5, p. 1275–1293, 2013. Citations on pages 66, 88, 89, and 90.
EKLUND, U.; BOSCH, J. Architecture for embedded open software ecosystems. *Journal of Systems and Software*, Elsevier, v. 92, p. 128–142, 2014. Citation on page 65.

ENARD, Q.; GATTI, S.; BRUNEAU, J.; MOON, Y.-J.; BALLAND, E.; CONSEL, C. Design-driven development of dependable applications: A case study in avionics. In: SCITEPRESS. *PECCS-3rd International Conference on Pervasive and Embedded Computing and Communication Systems*. [S.l.], 2013. Citations on pages 66, 67, and 88.

ESCOBAR, J.; LOSAVIO, F.; ORTEGA, D. Standard quality model to enterprise architecture support tools. In: *XXXIX Latin America Computing Conference (CLEI)*. Naiguata, VEN: [s.n.], 2013. p. 1–12. Citation on page 56.

ETEMAADI, R.; LIND, K.; HELDAL, R.; CHAUDRON, M. R. Quality-driven optimization of system architecture: Industrial case study on an automotive sub-system. *Journal of Systems and Software*, Elsevier, v. 86, n. 10, p. 2559–2573, 2013. Citations on pages 66, 88, and 92.

FAIRBANKS, G. *Just Enough Software Architecture: A Risk-Driven Approach*. [S.l.]: Marshall & Brainerd, 2010. Citation on page 36.

FEITOSA, D.; AMPATZOGLOU, A.; AVGERIOU, P.; NAKAGAWA, E. Y. Investigating Quality Trade-offs in Open Source Critical Embedded Systems. In: *11th International ACM SIGSOFT Conference on Quality of Software Architectures - QoSA (2015)*. Montreal, QC, Canada: [s.n.], 2015. p. 113–122. Citation on page 51.

FENTON, N. E.; PFLEEGER, S. L. *Software Metrics: A Rigorous and Practical Approach*. 2nd. ed. [S.l.]: PWS Publishing Co., 1998. Citations on pages 21 and 52.

FERNANDES, L. C.; SOUZA, J. R.; SHINZATO, P. Y.; PESSIN, G.; MENDES, C. C. T.; OSORIO, F. S.; WOLF, D. F. Intelligent Robotic Car for Autonomous Navigation: Platform and System Architecture. In: *2nd Brazilian Conference on Critical Embedded Systems*. Campinas, São Paulo, BRA: [s.n.], 2012. p. 12–17. Citation on page 26.

FLORENTZ, B.; HUHN, M. Architecture potential analysis: a closer look inside architecture evaluation. *Journal of Software*, v. 2, n. 4, p. 43–56, 2007. Citation on page 65.

FORSBERG, K.; MOOZ, H. The relationship of system engineering to the project cycle. In: *1st INCOSE International Symposium*. Chattanooga, TN: Wiley Online Library, 1991. p. 57–65. Citation on page 32.

FRANCH, X.; CARVALLO, J. P. A quality-model-based approach for describing and evaluating software packages. In: *Joint International Conference on Requirements Engineering*. Essen, DE: [s.n.], 2002. p. 104–111. Citation on page 56.

---

GARVIN, D. A. What Does “Product Quality” Really Mean? *Sloan Management Review*, v. 26, n. 1, p. 25–43, 1984. Citation on page 47.
GILL, H. Challenges for Critical Embedded Systems. In: 10th International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2005). Arlington, VA, USA: [s.n.], 2005. p. 7–9. Citations on pages 25, 35, and 60.

GILLIES, A. C. Software Quality - Theory and management. [S.l.]: Chapman & Hall, 1992. Citation on page 54.

GRADY, R.; CASWELL, D. Software Metrics: Establishing a Company-wide Program. [S.l.]: Prentice-Hall, 1987. Citation on page 55.

HAIR, J. F. Multivariate Data Analysis. [S.l.]: Prentice Hall, 2010. Citation on page 118.

HEESCH, U. V.; GERMANY, C.; ELORANTA, V.; AVGERIOU, P.; KOSKIMIES, K.; HARRISON, N. Decision-Centric Architecture Reviews. IEEE Software, v. 31, n. 1, p. 69–76, 2014. Citations on pages 22, 37, 42, 44, and 46.

HOFMEISTER, C.; KRUCHTEN, P.; NORD, R. L.; OBBINK, H.; RAN, A.; AMERICA, P. Generalizing a Model of Software Architecture Design from Five Industrial Approaches. In: 5th Working IEEE/IFIP Conference on Software Architecture. Washington: IEEE Computer Society, 2005. p. 77–88. Citations on pages 38, 39, and 42.

IEEE. IEEE Standard for a Software Quality Metrics Methodology. New York, NY, USA: [s.n.], 2009. Citation on page 52.

IEEE 1474.1. IEEE Standard for Communications-Based Train Control (CBTC) Performance and Functional Requirements. 2004. Citation on page 35.

INSAURRALDE, C. C.; SEMINARIO, M. A.; JIMÉNEZ, J. F.; GIRON-SIERRA, J. M. Model-based design analysis of an avionics fuel distributed control system. In: IEEE. 2008 IEEE/AIAA 27th Digital Avionics Systems Conference. [S.l.], 2008. p. 5–C. Citations on pages 66, 87, 88, and 90.

ISO/IEC 25000. Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. 2005. Citation on page 53.

ISO/IEC 25010. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. 2011. Citations on pages 22, 48, 54, 55, 68, 77, 78, 79, 93, 101, 102, 117, and 119.

ISO/IEC 9126. Software Engineering - Product quality, Part 1: Quality model. 2001. Citations on pages 22, 48, 53, 55, 56, 67, 68, and 78.

ISO/IEC 9126-2. Software Engineering - Product quality, Part 2: External metrics. 2001. Citation on page 53.

ISO/IEC/IEEE 24765. Systems and software engineering - Vocabulary. 2010. Citations on pages 21 and 68.

ISO/IEC/IEEE 42010. Systems and software engineering - Architecture description. 2010. Citation on page 39.

JANUZAJ, V.; KUGELE, S.; LANGER, B.; SCHALLHART, C.; VEITH, H. New Challenges in the Development of Critical Embedded Systems - An “aeromotive” Perspective. In: 4th International Conference on Leveraging Applications of Formal Methods, Verification, and Validation. Berlin, Heidelberg, DEU: [s.n.], 2010. p. 1–2. Citation on page 25.
JEONG, H. Y.; KIM, Y. H. A Quality Model of Lightweight Component for Embedded System. Application Mechanics and Materials, v. 121, p. 4907–4911, 2011. Citations on pages 22, 62, and 63.

KASUNIC, M. Designing an Effective Survey. [S.l.], 2005. 133 p. Citations on pages 97, 98, and 100.

KAZMAN, R.; ASUNDI, J.; KLEIN, M. Quantifying the Costs and Benefits of Architectural Decisions. In: 23rd International Conference on Software Engineering (ICSE 2001). Washington, DC, USA: IEEE Computer Society, 2001. p. 297–306. Citation on page 37.

KAZMAN, R.; KLEIN, M.; CLEMENTS, P. ATAM: Method for Architecture Evaluation. [S.l.], 2000. 83 p. Citations on pages 37, 42, and 43.

KHARCHENKO, A.; GALAY, I.; YATCYSHYN, V. The method of quality management software. In: 7th International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH). Polyana, RUS: [s.n.], 2011. p. 82–84. Citation on page 56.

KIM, C.; LEE, K. Software Quality Model for Consumer Electronics Product. In: Quality Software, 2009. QSIC '09. 9th International Conference on. Ieee, 2009. p. 390–395. ISBN 1550-6002 VO -. ISSN 1550-6002. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5381393>. Citation on page 77.

KITCHENHAM, B. Towards a constructive quality model. Part 1: Software quality modelling, measurement and prediction. Software Engineering Journal, v. 2, n. 4, p. 105–126, 1987. Citation on page 55.

KITCHENHAM, B.; PICKARD, L. M. Towards a constructive quality model. Part 2: Statistical techniques for modelling software quality in the ESPRIT REQUEST project. Software Engineering Journal, v. 2, n. 4, p. 114–126, 1987. Citation on page 55.

KLÄS, M.; LAMPASONA, C.; MÜNCH, J. Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results. In: 37th EUROMICRO Conference on Software Engineering and Advanced Applications. Oulu, FIN: [s.n.], 2011. p. 341–348. Citations on pages 56 and 77.

LEE, K. C.; JEON, J. W.; HWANG, D. H.; KIM, Y. J. Implementation and tests of anti-lock braking algorithm for a heavy vehicle. In: International Conference on Control Automation and Systems. [S.l.: s.n.], 2002. Citation on page 34.

LOCHMANN, K.; GOEB, A. A unifying model for software quality. In: . New York, New York, USA: ACM Press, 2011. p. 3. ISBN 9781450308519. Available: <http://dl.acm.org/citation.cfm?doid=2024587.2024591>. Citation on page 77.

LONIEWSKI, G.; BORDE, E.; BLOUIN, D.; INSFRAN, E. Model-driven requirements engineering for embedded systems development. In: IEEE. 2013 39th Euromicro Conference on Software Engineering and Advanced Applications. [S.l.], 2013. p. 236–243. Citations on pages 66, 87, and 88.

LOSAVIO, F.; CHIRINOS, L.; PÉREZ, M. A. QUALITY MODELS TO DESIGN SOFTWARE ARCHITECTURES. In: Technology of Object-Oriented Languages and Systems. [S.l.: s.n.], 2001. p. 1–13. Citation on page 67.
LUZEAUX, D.; RUAULT, J. Systems of Systems. [S.l.]: Wiley, 2010. Citation on page 91.

MAIER, M. W. Architecting principles for systems-of-systems. Systems Engineering, v. 1, p. 267–284, 1998. Citation on page 91.

MALEK, S.; EDWARDS, G.; BRUN, Y.; TAJALLI, H.; GARCIA, J.; KRKA, I.; MEDVIDOVIC, N.; MIKIC-RAKIC, M.; SUKHATME, G. S. An architecture-driven software mobility framework. Journal of Systems and Software, Elsevier, v. 83, n. 6, p. 972–989, 2010. Citations on pages 65 and 88.

MARTENSSON, F.; GRAHN, H.; MATTSSON, M. Evaluating software quality attributes of communication components in an automated guided vehicle system. In: IEEE. 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS’05). [S.l.], 2005. p. 550–558. Citations on pages 66, 88, 90, and 91.

MARWEDEL, P. Embedded System Design - Embedded Systems Foundations of Cyber-Physical Systems. 2nd. ed. [S.l.]: Springer, 2011. Citations on pages 21, 25, 27, 28, 29, 30, 31, 32, 33, 34, and 35.

MCCALL, J. A.; RICHARDS, P. K.; WALTERS, G. Factors in software quality. [S.l.], 1977. Citations on pages 21, 22, 48, 55, 56, 67, and 77.

MEEDENIY, I.; ALETI, A.; MOSER, I. Efficient sensitivity analysis of reliability in embedded software. In: SPRINGER. Australasian Conference on Artificial Life and Computational Intelligence. [S.l.], 2015. p. 395–408. Citations on pages 66, 87, and 88.

MEEDENIY, I.; BUHNOVA, B.; ALETI, A.; GRUNSKE, L. Architecture-driven reliability and energy optimization for complex embedded systems. In: SPRINGER. QoSA. [S.l.], 2010. p. 52–67. Citations on pages 66, 87, and 88.

NAKAGAWA, E. Y.; OQUENDO, F.; BECKER, M. RAModel: A Reference Model for Reference Architectures. In: 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture. [S.l.: s.n.], 2012. p. 297–301. Citation on page 94.

NIU, N.; Da Xu, L.; CHENG, J. C.; NIU, Z. Analysis of Architecturally Significant Requirements for Enterprise Systems. IEEE Systems, v. 8, n. 3, p. 850–857, 2014. Citation on page 37.

NOERGAARD, T. Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers. [S.l.]: Elsevier Science, 2012. Citations on pages 27, 28, 30, and 31.

OLIVEIRA, L. B. R.; GUESSI, M.; FEITOSA, D.; MANTEUFFEL, C.; GALSTER, M.; OQUENDO, F.; NAKAGAWA, E. Y. An Investigation on Quality Models and Quality Attributes for Embedded Systems. In: 8th International Conference on Software Engineering Advances (ICSEA 2013). Venice, ITA: [s.n.], 2013. p. 523–528. Citations on pages 21 and 22.

ORTIZ, F.; ÁLVAREZ, B.; PASTOR, J. Á.; SÁNCHEZ, P. A case study in performance evaluation of real-time teleoperation software architectures using uml-mast. In: SPRINGER. International Conference on Reliable Software Technologies. [S.l.], 2003. p. 417–428. Citations on pages 65 and 88.

PAULITSCH, M.; RUSS, H.; SOREA, M. Non-functional avionics requirements. In: Leveraging Applications of Formal Methods, Verification and Validation. Porto Sani, Greece: Springer Berlin Heidelberg, 2008. p. 369–384. Citations on pages 26 and 34.
PEARSE, T.; OMAN, P. Maintainability measurements on industrial source code maintenance activities. In: Proceedings of International Conference on Software Maintenance. [S.l.: s.n.], 1995. p. 295–303. Citation on page 73.

PRESSMAN, R. Software Engineering - A Practitioner’s Approach. 7th ed. [S.l.]: McGraw-Hill, 2010. Citation on page 32.

RADJENOVIC, A.; PAIGE, R. Architecture description languages for high-integrity real-time systems. IEEE Software, IEEE, v. 23, n. 2, p. 71–79, 2006. Citations on pages 65 and 87.

REDWINE JR., S. T.; RIDDLE, W. E. Software technology maturation. In: 8th International Conference on Software Engineering (ICSE 1985). Los Alamitos, CA, USA: IEEE Computer Society Press, 1985. p. 189–200. Citation on page 37.

RODRIGUES, D. Using SOA in critical-embedded systems. In: International Conference on Internet of Things (iThings/CPSCOM) and 4th International Conference on Cyber, Physical and Social Computing. Dalian, CH: [s.n.], 2011. p. 733–738. Citation on page 21.

ROLIA, J. A.; SEVCIK, K. C. The method of layers. IEEE Transactions on Software Engineering, IEEE, v. 21, n. 8, p. 689–700, 1995. Citation on page 73.

SADDEM, R.; TOGUYENI, A.; TAGINA, M. Diagnosis of critical embedded systems: application to the control card of a railway vehicle braking systems. In: International Conference on Automation Science and Engineering (CASE 2011). Trieste, ITA: [s.n.], 2011. p. 163–168. Citation on page 25.

SEO, C.; MALEK, S.; MEDVIDOVIC, N. Component-level energy consumption estimation for distributed java-based software systems. In: International Symposium on Component-Based Software Engineering (CBSE2008). Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 97–113. Citations on pages 72 and 73.

SHAW, M.; CLEMENTS, P. The golden age of software architecture. IEEE Software, v. 23, n. 2, p. 31–39, 2006. Citations on pages 37 and 38.

SILVA, N.; VIEIRA, M. Certification of embedded systems: Quantitative analysis and irrefutable evidences. In: International Symposium on Software Reliability Engineering Workshops (ISSREW). Pasadena, CA: [s.n.], 2013. p. 15–16. Citation on page 21.

SINGH, B.; KANNOJIA, S. P. A Review on Software Quality Models. In: International Conference on Communication Systems and Network Technologies (CSNT 2013). Gwalior, Madhya Pradesh, IND: [s.n.], 2013. p. 801–806. Citation on page 54.

SOMMERVILLE, I. Software Engineering. 9th ed. [S.l.]: Addison-Wesley, 2010. Citation on page 47.

TRAON, Y. L.; OUABDESELAM, F.; ROBACH, C. Analyzing testability on data flow designs. In: IEEE. Software Reliability Engineering, 2000. ISSRE 2000. Proceedings. 11th International Symposium on. [S.l.], 2000. p. 162–173. Citations on pages 65 and 90.

VILELA, J.; FIGUEIREDO, B.; CASTRO, J.; SOARES, M.; GONÇALVES, E. Usability and software architecture: A literature review. In: 2015 IX Brazilian Symposium on Components, Architectures and Reuse Software. [S.l.: s.n.], 2015. p. 80–89. Citation on page 89.
VILLALBA, M. T.; FERNÁNDEZ-SANZ, L.; MARTÍNEZ, J. J. Empirical support for the generation of domain-oriented quality models. *IET Software*, v. 4, n. 1, p. 1–14, 2010. Citations on pages 56, 58, 77, and 103.

WAGNER, S. *Software Product Quality Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. Citations on pages 54, 67, and 77.

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)*. Zürich, CHE: [s.n.], 2012. p. 1133–1142. Citations on pages 22, 52, 53, 54, and 55.

WIJNSTRA, J. G. Quality attributes and aspects of a medical product family. In: *34th Annual Hawaii International Conference on System Sciences (HICSS 2001)*. Washington, DC, USA: [s.n.], 2001. p. 1–10. Citations on pages 26 and 73.

_____. From problem to solution with quality attributes and design aspects. *Journal of Systems and Software*, Elsevier, v. 66, n. 3, p. 199–211, 2003. Citations on pages 66, 87, 88, 89, 90, and 92.

XIONG, X.; LIU, J.; ZHANG, M.; DING, Z. Modeling and verification of an automatic train protection system. In: *International Computer Software and Applications Conference*. Soul, KR: [s.n.], 2010. p. 226–231. Citation on page 35.

XU, T.; LIU, Z.; TANG, T.; ZHENG, W.; ZHAO, L. Component based design of fault tolerant devices in cyber physical system. In: IEEE. *Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2012 15th IEEE International Symposium on*. [S.l.], 2012. p. 37–42. Citations on pages 66 and 87.
SYSTEMATIC MAPPING PROTOCOL
Software Quality in the context of Critical Embedded Systems:
"Systematic Literature Review"

Brauner R. N. Oliveira*, Daniel S. S. Santos*, Dênis L. Zaniro*, Lina M. Garcés Rodríguez*, Apostolos Ampatzoglou†, Paris Avgeriou†, and Elisa Y. Nakagawa*.

October 7, 2014

---

* Instituto de Ciências Matemáticas e de Computação (ICMC) - Universidade de São Paulo (USP)
† Department of Computer Science - University of Groningen (RuG)
CONTENTS

1 Introduction 3

2 Systematic Literature Review Protocol 4
   2.1 SLR Objective ............................................................... 4
   2.2 Question Quality and Amplitude ..................................... 5
   2.3 Source selection ............................................................ 5
      2.3.1 Sources selection criteria definition ......................... 6
      2.3.2 Studies languages .................................................... 6
      2.3.3 Sources identification .............................................. 7
   2.4 Study Selection ............................................................ 8
      2.4.1 Studies Inclusion and Exclusion Criteria Definition ........ 8
   2.5 Procedures for Studies Selection .................................... 9
   2.6 Data Extraction Strategy ............................................... 10
   2.7 Synthesis of the Extracted Data ..................................... 12

3 Protocol Validation 12

4 Report the Review 12
1 Introduction

Embedded systems are one of the most important types of software-intensive systems, as they are being used by almost everyone in the modern society, being part of the peoples’ life more than any other type of software-intensive systems. For instance, they are found in mobile phones, cars, fabrication and medicine equipments, etc. Some of those systems are considered as critical systems (i.e., Critical Embedded Systems - CES), due to their failure may cause serious damage to human lives and to the environment, damage to expensive equipment or unrecoverable financial losses. In this context, the establishment of quality attributes, characteristics and metrics for CES is important to avoid critical damages. In this perspective, Quality Models (QM) have been used to describe, manage, and predict software quality [1]. QM categorize quality attributes (QA) in such a way that it is possible to identify their relations, through an hierarchical approach [2]. Shortly, a QA is defined as a characteristic of software that affects the quality of the system [2]. Moreover, QA can be defined using metrics, i.e., a quantitative measure of the degree to which an item possesses a given quality attribute [2]. Furthermore, the establishment of QA and their related metrics for a software systems is an important basis when architectural decisions are made. In this context, the selection of architectural styles, patterns, technologies, components, viewpoints, or views, that could compose a software architecture, must be selected in order to address the established QA for that systems [3].

Despite the importance of having a broad knowledge about software quality for CES, there is a lack of a complete and detailed analysis of the state of the art in this research topic. In this context, a systematic literature review (SLR) will be conducted aiming to obtain a deep understanding on how software quality in the context of CES has been addressed. Specifically, we will focus on QM, QA, and quality metrics that have been proposed to establish and evaluate CES. For this, we will follow the guidelines proposed by Kitchenham and Charters [4]. This process provides a reliable means and establishes methods for conducting a comprehensive literature review based on three clearly defined phases: (i) planning; (ii) conducting; and (iii) reporting the review. During the planning phase, the need for a review is identified and the review protocol is developed. In the review protocol, the following items are specified: (i) Question formularization; (ii) Source selection; (iii) Study selection; and (iv) Procedures for studies selection, among others. In the conducting phase, all the tasks that were defined in the protocol are executed. Firstly, the primary studies are selected according to the inclusion criteria. Secondly, the quality of these studies is evaluated and, finally, all the relevant data are extracted and synthesized. Finally, in the reporting phase, the results of the review are reported. Moreover, this SLR will be conducted in the frame of an international research project titled “Empirical Software Engineering for Critical Embedded Systems (ESECES)”, that involves researchers of the Institute of Mathematics and Computer Sciences (ICMC) at University of São Paulo (USP), and of the University of Groningen (RuG). The principal goal of this project is to develop a set of reusable architecture design artifacts; QM for critical QA; an architectural
design method; and an architectural evaluation method.

In the remainder of this document is detailed the protocol of this SLR.

2 Systematic Literature Review Protocol

2.1 SLR Objective

To accomplish the SLR goal, we have defined four research questions with some sub-questions:

RQ1: What are the QM established for CES domain?
   - RQ1.1: How have these QM been defined (e.g., based on standards as ISO/IEC 25010, based on accumulated experience, or based on metamodels)?
   - RQ1.2: Which purpose the QM has (i.e., description, assessment, or prediction)?
   - RQ1.3: How has been the QM evaluated (e.g., using prototypes or software systems)?
   - RQ1.4: Which elements comprise the QM?

RQ2: What are the most important QA for CES?
   - RQ2.1: Which are the QA established for the different CES sub-domains?
   - RQ2.2: How the QA were defined (e.g., using domain-specific or generic QM, or based on accumulated experience)?

RQ3: Which metrics have been used to measure the QA?
   - RQ3.1: Which metrics were established for the QA?
   - RQ3.2: How these metrics were defined?
   - RQ3.3: How have been the metrics evaluated (e.g., using prototypes or real systems)?

RQ4: Which architectural decisions (e.g., selection of architectural styles or patterns) have been made to achieve QA on CES?

Research questions overview

The RQ1 aims at finding QM that were proposed to the domain of CES, and how they were defined, i.e., we want to find out if they were defined based on a process, standards (e.g., ISO SQuaRE [5]) or a custom method. Furthermore, we intend to classify these models using the approach presented by Deissenboeck et al. [6]. That approach classify the QM according to three main purposes: 1) defining quality; 2) assessing the quality; and 3) predicting quality. Finally, we will perform an evaluation to determine the completeness of
the QM elements (e.g., characteristic, properties, etc. that they contain), considering the methodology for building such models proposed by Franch and Carvallo [7].

The RQ2 aims at finding which QA have been discussed under the context of CES, if there are differences in the definition of QA among different sub-domains, (e.g., modifiability for robotic systems may be composed by different attributes when compared to other domains, such as medical systems), and if their definition were based on existing QM.

The RQ3 is complementary to RQ2, since its goals are to find metrics to measure the QA previously found, trace a relation between metrics and QA, find out how these metrics were defined, and the degree of success to use that metrics in measuring such QA.

Considering that system architecture is critical to the realization of many qualities of interest in a system, the RQ4 aims at finding which architectural strategies have been used to achieve the QA found in RQ2.

2.2 Question Quality and Amplitude

We have also defined the PICO (Population, Intervention, Comparison and Outcomes)[8] for our SLR, presented as follow:

**Population:** It identifies the population group observed by the intervention. The population of our SLR is: “Studies that address QM, QA, and quality metrics in the CES domain”.

**Intervention:** It refers to what will be observed in the context of the SLR. The intervention of this SLR is represented by the software quality concepts, specifically QA, QM, and quality metrics.

**Comparison:** It defines what is compared in the context of the SLR. It is possible to compare different QA and QM, both in different application domains.

**Outcomes:** It expresses the results of the SLR. Particularly, considering our SLR, the main result is the analysis and characterization of the studies proposed in the literature, that report QA, QM, and Quality metrics.

2.3 Source selection

In this section, we have selected the sources where searches for primary studies will be executed.
2.3.1 Sources selection criteria definition

Here we have defined which criteria will be used to evaluate studies sources, i.e., which characteristics make these sources candidate to be used in the SLR execution. There are three types of sources that will be considered in this SLR:

- **Publication databases**: Aiming at establishing which publication databases will be used to find the primary studies, we have used the following criteria [9]: *coverage* (large number of conferences proceedings, journals, and indexed books in different knowledge areas), *content update* (publications are regularly updated); *availability* (full text of the papers is 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). The databases selected are presented in Table 2.1:

  | Source            | URL                               |
  |-------------------|-----------------------------------|
  | ACM Digital Library | http://dl.acm.org/               |
  | Compendex         | http://www.engineeringvillage.com/|
  | IEEE Xplore       | http://ieeexplore.ieee.org/Xplore/home.jsp |
  | ScienceDirect     | http://www.sciencedirect.com/     |
  | Scopus            | http://www.scopus.com/            |
  | SpringerLink      | http://link.springer.com/         |
  | Web of Science    | http://apps.webofknowledge.com/   |

- **Related works**: We will also consider the snowball technique proposed by Ridley [10] to cover the whole research area. This technique will allow to identify and examine research works cited in the studies selected.; and

- **Specialist**: We will also consider studies suggested by specialists of CES and software architecture domains. Although the indication of studies by specialists can be considered as bias, we will adopt this source aiming not to lose any important evidence.

2.3.2 Studies languages

As the chosen language may restrain the sources identification, we have defined the languages in which obtained primary studies must be written. Considering this SLR, the defined language is English, because most of research in Computer Science have been reported in this language.
2.3.3 Sources identification

This item aims at selecting sources for the review execution. We have defined the methods used to execute the search for primary studies. In addition, we have identified the keywords and the search string that will be used to search primary studies.

**Primary study search methods:** The search aims at the identification of candidate primary studies that answer the research questions and that would be either included or excluded to the final set of review studies.

We will use automatic search. By automatic search, we mean a search performed by executing search strings on search engines of publication databases. The search string that we will use is presented in the Table 2.2.

**Keywords:** The main keywords are “Critical Embedded System”, “Quality model”, “Quality attribute”, and “Quality metric”. Following, the main keywords and their related terms are:

- **Critical Embedded System:** Critical Embedded (Software | Computer System), Real-Time System, Embedded Real-Time, Critical Real-Time, and Real-Time (Embedded System | Embedded Software)

- **Quality Model:** Quality Taxonomy.

- **Quality Attribute:** Non-functional (Requirement | Property | Characteristic), and Quality (Requirement | Factor | Characteristic | Criteria | Criterion).

- **Quality Metric:** Quality (Measure | Measurement).

**Search string:** It combine keywords and their synonymous arranged in such a way that the highest amount of relevant studies is obtained from search engines. For each query part, a list of alternative keywords will be used and connected through logical OR to form a more expressive query. The search string is presented in the Table 2.2.

**Sources list:** The publication databases selected for our SLR are shown in Table 2.1. According to Dybå et al. [11], these databases are efficient to conduct SLR in the context of Software Engineering. Furthermore, Scopus has been added, since it is considered the largest database of abstracts and citations [4];
2.4 Study Selection

In this section we describe the process and the criteria for studies selection and evaluation.

2.4.1 Studies Inclusion and Exclusion Criteria Definition

This subsection presents the criteria by which studies will be evaluated to decide if they must be selected or not in the context of the SLR. The Inclusion Criteria (IC) or Exclusion Criteria (EC) make it possible to include primary studies that are relevant to answer the research questions and exclude studies that do not answer them. The IC/EC will be used in every stage are listed as follows:

Inclusion Criteria (IC):

IC1 The study proposes/studies a QM for CES;

IC2 The study address QA in the context of CES; and
IC3 The study address quality metrics for CES QA.

Exclusion criteria (EC):

EC1 The study proposes a QM, QAs or quality metrics but does not address CES;
EC2 The study does not propose a QM, QAs, or quality metrics but address CES;
EC3 The study does not propose a QM, QAs, or quality metrics neither address CES;
EC4 The study is a book, editorial, keynote, opinion, tutorial, poster or panel;
EC5 The study is a previous version of a more complete study about the same research;
EC6 The paper language is different from English;
EC7 The full paper is not available; and
EC8 The study proposal is out of our review goal.

2.5 Procedures for Studies Selection

Aligned with Dybä and Dingsøyr [12], there are four stages of filtering the article set to produce the primary study data set.

Stage 1. First Selection

1. To debug the primary studies set, we will make a selection of the papers that are repeated in different databases. This depuration will be made with JabRef Reference Manager tool.

2. Based on the debugged set of studies, the title, abstract and conclusions, if necessary, of each primary study will be read, and the inclusion and exclusion criteria will be applied.

The studies in this selection phase will be divided to four members of the team (two Master and two Ph.D. students) and possible doubts will be resolved by consensus with their supervisor (reference architecture specialist). We will not apply an agreement measure because the number of researchers involved in the review is not significantly large.

For each data source mentioned in Table 2.1, we will document the number of papers that have been returned. In addition, we will record the number of papers left for each venue after primary study selection on the basis of title and abstract. Moreover, the number of papers finally selected from each source will be recorded.

Stage 2. Second Selection
1. Each primary study selected will be read in full and analyzed according to inclusion and exclusion criteria.

2. In addition, the related works cited by these studies will be evaluated intending to cover the whole research area.

3. If the decision about the inclusion or exclusion of a study is not clear, this study will be analyzed by the supervisor (specialist). When a disagreement occurs, consensus will be conducted.

4. As in Stage 1, the number of papers finally selected from each source will be recorded.

**Stage 3. Quality Assessment**

All selected papers which were included in the review will undergo a quality assessment. Thus, each study will be evaluated against a set of questions with regard to the used method and the quality of the reporting.

We will adopt the quality assessment instrument used by Dyba and Dingsoyr [12]. This instrument uses a three point scale to answer each question, either as “yes”, “to some extent” or “no”. By including “to some extent” we do not neglect statements where authors provided only limited information to answer the assessment questions. Each quality assessment question will be answered by assigning a numerical value (1 = “yes”, 0 = “no”, and 0.5 = “to some extent”). Then, a quality assessment score will be given to a study by summing up the scores for all the questions for a study (quality assessment score of a study).

Quality criteria are:

**Q1:** Is there a rationale for why the study was undertaken?

**Q2:** Is there an adequate description of the context (i.e., home, work or community) in which the research was carried out?

**Q3:** Is there a description of the quality requirements?

**Q4:** Is there a justification and description for the research design?

**Q5:** Does the study provide an evaluation of the research design?

**Q6:** Do the authors discuss the credibility of their findings?

**Q7:** Are limitations of the study discussed explicitly?

Each quality question score for each primary study will be register in the Table 2.3.

**2.6 Data Extraction Strategy**

During the data collection phase, we will collect a set of variables that describe each primary study. Data collection will be handled by the students and their supervisor. Every
researcher involved will complete data extraction for every primary study separately. Then the three values of each variable (obtained by each researcher) will be compared to each other and the final value of the variable will be assigned to the primary study after discussion on every researcher’s opinion. If three or more researchers assigned the same value to one variable this value will be assign to the variable without further discussion. In any other case, after a debate among the authors, a value will be assigned to every variable. We will not apply an agreement measure because the number of researchers involved in the review is not significantly large. However, all conflicts will be recorded. The Table 2.4 presents the data that will be extracted from every study.

| V01. Author    | V11. QM purpose |
| V02. Year      | V12. QM evaluation |
| V03. Title     | V13. QA metrics |
| V04. Source    | V14. Metric definition |
| V05. Keywords  | V15. Metric evaluation |
| V06. Application domain | V16. Architectural decisions |
| V07. QA        | V17. Aimed QA |
| V08. QA definition process | V18. Author’s organization |
| V09. QM        | V19. Relevant references |
| V10. QM definition process | V20. Comments |
2.7 Synthesis of the Extracted Data

Relationships between extracted variables and research questions are provided in the Table 2.5, accompanied by the synthesis or analysis methods used on the data.

| Research Question | Variables Used       | Synthesis Method                                                  |
|-------------------|----------------------|-------------------------------------------------------------------|
| RQ1               | [V09, V10, V11, V12] | Descriptive statistics and descriptive synthesis.                |
| RQ2               | [V07, V08]           | Descriptive statistics and descriptive synthesis.                 |
| RQ3               | [V13, V14, V15]      | Descriptive statistics and descriptive synthesis.                 |
| RQ4               | [V16, V17]           | Descriptive statistics and descriptive synthesis.                 |

3 Protocol Validation

The SLR will be validated in three parts:

1. The pilot → testing the process:
   Researchers will use a subset of resources to test the process. Problems in replicating the process may be identified, and the process will be refined accordingly.
   Gaps in searches may be identified, and search terms and resources will be changed to include missing papers.

2. Data extraction:
   Reliability of how to extract details from papers will be tested. An independent researcher (not involved in the pilot) will be given a set of accepted papers and asked to fill in the data extraction form.

3. External reviewers will review the protocol.

4 Report the Review

When reporting our study, we will a) discuss findings (description of primary studies, results of any quantitative summaries, and details of any meta-analysis), b) present a discussion (principal findings, strengths and weaknesses of the review (also in relation to other reviews), and the meaning of findings, such as applicability of findings, benefits, adverse effects and risks), c) discuss conclusions, including recommendations, such as practical implications for software development, implications for practitioners, and unanswered questions and implications for future research, and d) elaborate on limitations of the review.
REFERENCES

[1] L. B. R. Oliveira, M. Guessi, D. Feitosa, C. Manteuffel, M. Galster, F. Oquendo, and E. Y. Nakagawa, “An Investigation on Quality Models and Quality Attributes for Embedded Systems”, in The Eighth International Conference on Software Engineering Advances (ICSEA 2013), Venice, ITA, October 2013, pp. 523–528.

[2] ISO/IEC/IEEE, “ISO/IEC/IEEE 24765:2010 Systems and software engineering - Vocabulary”, 2010.

[3] Len Bass, Paul Clements, and Rick Kazman, Software Architecture in Practice, Addison-Wesley, 2nd edition, 2003.

[4] B. A. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001.”, Tech. Rep., Keele University and Durham University, UK, 2007.

[5] ISO/IEC, “ISO/IEC 25010:2014: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE”, 2014.

[6] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner, “Software quality models: Purposes, usage scenarios and requirements”, in Workshop on Software Quality (WOSQ 2009), Vancouver, BC, 2009, pp. 9–14.

[7] X. Franch and J. P. Carvallo, “A quality-model-based approach for describing and evaluating software packages”, Essen, North Rhine-Westphalia, DEU, 2002.

[8] J. Biolchini, P. Mian, A. Natali, and G. Travassos, “Systematic Review in Software Engineering. Technical report (TRES679/05)”, Tech. Rep., PESC - COPPE/UFRJ, 2005.

[9] O. Dieste, A. Grimán, and N. Juristo, “Developing search strategies for detecting relevant experiments”, Empirical Software Engineering, vol. 14, no. 5, pp. 513–539, 2009.

[10] D. Ridley, The Literature Review: A Step-by-Step Guide for Students, SAGE Study Skills Series. SAGE Publications, 2012.

[11] T. Dybå, T. Dingsoyr, and G. K. Hanssen, “Applying systematic reviews to diverse study types: An experience report”, in First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), Washington, DC, USA, 2007, pp. 225–234, IEEE Computer Society.

[12] T. Dybå and T. Dingsoyr, “Strength of evidence in systematic reviews in software engineering”, in Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2008), New York, NY, USA, 2008, ESEM ’08, pp. 178–187, ACM.
QUESTIONNAIRE FOR QM4CES EVALUATION
Quality Model (QM4CES) Validation

Thanks for accepting my invitation! This survey should take no more than 30 minutes.

SURVEY DESCRIPTION

During software architecture design, the most important quality attributes regarding the intended software system must be managed. In this scenario, this survey is the final stage of a Master's project that aims the establishment of QM4CES, a quality model (QM) for Critical Embedded Systems (CES). QM4CES was proposed based on evidence found at conducting a Systematic Mapping. This quality model also supports architectural design activities performed to critical embedded systems that must avoid economic or physical damage when they are under operation. The goal of this survey is to understand which of the quality attributes proposed by QM4CES are considered the most important ones by domain experts and the degree of importance to develop CES.

This survey is composed of four parts:

1. Participant Information: Information about yourself and your professional experience;
2. Quality Attributes: you will evaluate the set of quality attributes contained in QM4CES;
3. Quality attribute trade-offs: you will identify trade-off relation between quality attributes; and
4. Final Considerations: you will answer about the utility and completeness of QM4CES. You can also submit additional information that you consider important.

More information about these parts are presented within their sections.

Survey dates

1st reminder: January 24, 2017
2nd reminder: January 30, 2017
Deadline: January 31, 2017

Tip: you can save your answers at any time and resume later.

Confidentiality Remarks: All information provided by you in this survey is completely confidential. It will be only used in this study and will not be analyzed individually within information provided by other participants.

This research is supported by the brazilian agencies CNPq (http://cnpq.br/) and CAPES (https://www.capes.gov.br/).

There are 28 questions in this survey

Participant Information

Provide some information about you

Age  *

Choose one of the following answers

Please choose only one of the following:

- less than 21 years
- between 22 and 30 years
- between 31 and 40 years
- between 41 and 50 years
- 51 years or more
Country of residence  *

Choose one of the following answers

Please choose only one of the following:

○ Afghanistan
○ Albania
○ Algeria
○ Andorra
○ Angola
○ Antigua and Barbuda
○ Argentina
○ Armenia
○ Australia
○ Austria
○ Azerbaijan
○ Bahamas
○ Bahrain
○ Bangladesh
○ Barbados
○ Belarus
○ Belgium
○ Belize
○ Benin
○ Bhutan
○ Bolivia
○ Bosnia and Herzegovina
○ Botswana
○ Brazil
○ Brunei
○ Bulgaria
○ Burkina Faso
○ Burundi
○ Cabo Verde
○ Cambodia
○ Cameroon
○ Canada
○ Central African Republic (CAR)
○ Chad
○ Chile
○ ...
Education  *
Choose one of the following answers

Please choose only one of the following:

- Undergraduate student
- Bachelor’s degree
- Master’s student
- Master’s degree
- Ph.D. student
- Ph.D. degree

Choose the option that best fits to your education level

Current occupation  *

Check all that apply

Please choose all that apply:

- Undergraduate student
- Master’s student
- PhD. candidate
- Researcher
- Lecturer
- Professor
- Software Engineer
- Project Manager
- Other: [space for text]

- Please, select one or more application domains in which you have had research or practical experience in the development of CES.
- (Optional) For each application domain you selected, write the type of one or more systems you worked with, e.g., "flight control system", "anti-collision system", "autonomous navigation system".

* Comment only when you choose an answer.

Please choose all that apply and provide a comment:

- Automotive

- Avionics
If the application domain(s) of your expertise is(are) listed, please, choose the option “other”.

How many years do you have of experience developing or researching CES? *

Choose one of the following answers

Please choose only one of the following:

- less than 1 year
- between 1 and 3 years
- between 3 and 5 years
- more than 5 years

How do you classify your expertise with CES? *

Check all that apply

Please choose all that apply:

- Provider
- Domain analysis
- Design
- Development
- Testing
- Assessment
- Maintenance
- Academic Research
- Industrial Research
- Other:
Please, select architectural design phases you have experience. *

Check all that apply

Please choose all that apply:

- [ ] Architectural Analysis (architecture requirements)
- [ ] Architectural Synthesis (architecture documentation)
- [ ] Architectural Evaluation (architecture appropriateness)
- [ ] None
- [ ] Other: 

Please, select as many approaches below you have experience or are fully aware of their role in software development. *

Check all that apply

Please choose all that apply:

- [ ] ADD (Attribute-Driven Design)
- [ ] AIW (Architecture Improvement Workshop)
- [ ] ARID (Active Reviews for Intermediate Designs)
- [ ] ATAM (Architecture Tradeoff Analysis Method)
- [ ] CBAM (Cost Benefit Analysis Method)
- [ ] DCAR (Decision-Centric Architecture Reviews)
- [ ] MTW (Mission Thread Workshop)
- [ ] PALM (Pedigreed Attribute eLicitation Method)
- [ ] QAW (Quality Attribute Workshop)
- [ ] SAAM (Software Architecture Analysis Method)
- [ ] None

Quality Attributes of QM4CES

In this second questionnaire, quality attributes (i.e., properties or characteristics of software systems) are subject to judgment regarding their importance from the point of view of experts or stakeholders, and their influence on the software architecture of a CES, as stated before. Here, we present the set of attributes we have selected for our quality model. Please, answer each question below based on your experience with CES.

- RELIABILITY is composed by the following quality attributes:
  - Availability
  - Fault tolerance
  - Maturity
  - Recoverability
Would you be willing to take extra cost and effort to increase the degree to which the CES...

* 

Please choose the appropriate response for each item:

|                          | Never | Rarely | Possibly | Probably | Always |
|--------------------------|-------|--------|----------|----------|--------|
| is operational and accessible when required for use? (Availability) | ☐     | ☐      | ☐        | ☐        | ☐      |
| operates as intended despite the presence of hardware or software faults? (Fault tolerance) | ☐     | ☐      | ☐        | ☐        | ☐      |
| meets needs for reliability under normal operation? (Maturity) | ☐     | ☐      | ☐        | ☐        | ☐      |
| can recover the data directly affected and re-establish the desired state of the system in the event of an interruption or a failure? (Recoverability) | ☐     | ☐      | ☐        | ☐        | ☐      |

- PERFORMANCE EFFICIENCY is composed by the following quality attributes:
  - Capacity
  - Resource utilization
  - Time behavior

Would you be willing to take extra cost and effort to increase the degree to which the...

* 

Please choose the appropriate response for each item:

| maximum limits of the CES meet requirements? (Capacity) | Never | Rarely | Possibly | Probably | Always |
|---------------------------------------------------------|-------|--------|----------|----------|--------|
|                                                         | ☐     | ☐      | ☐        | ☐        | ☐      |
amounts and types of resources used by the CES, when performing its functions, meet requirements? (Resource utilization)

| Never | Rarely | Possibly | Probably | Always |
|-------|--------|----------|----------|--------|
| ○     | ○      | ○        | ○        | ○      |

response and processing times and throughput rates of the CES, when performing its functions, meet requirements? (Time behavior)

| Never | Rarely | Possibly | Probably | Always |
|-------|--------|----------|----------|--------|
| ○     | ○      | ○        | ○        | ○      |

- **USABILITY** is composed by the following quality attributes:
  - Learnability
  - Operability
  - User error protection

Would you be willing to take extra cost and effort to increase the degree to which the CES...

* Please choose the appropriate response for each item:

| Never | Rarely | Possibly | Probably | Always |
|-------|--------|----------|----------|--------|
| ○     | ○      | ○        | ○        | ○      |

enables the user to learn how to use it with effectiveness, efficiency in emergency situations? (Learnability)

| Never | Rarely | Possibly | Probably | Always |
|-------|--------|----------|----------|--------|
| ○     | ○      | ○        | ○        | ○      |

is easy to operate, control and appropriate to use? (Operability)

| Never | Rarely | Possibly | Probably | Always |
|-------|--------|----------|----------|--------|
| ○     | ○      | ○        | ○        | ○      |

protects users against making errors? (User error protection)

| Never | Rarely | Possibly | Probably | Always |
|-------|--------|----------|----------|--------|
| ○     | ○      | ○        | ○        | ○      |

- **MAINTAINABILITY** is composed by the following quality attributes:
  - Analyzability
  - Modifiability
  - Modularity
  - Testability
Would you be willing to take extra cost and effort to increase the...

* 

Please choose the appropriate response for each item:

|                                      | Never | Rarely | Possibly | Probably | Always |
|--------------------------------------|-------|--------|----------|----------|--------|
| degree of effectiveness and efficiency with which it is possible to assess the impact on the CES of an intended change to one or more of its parts, or to diagnose the CES for deficiencies or causes of failures, or to identify parts to be modified? (Analyzability) |       |        |          |          |        |
| degree to which the CES can be effectively and efficiently modified without introducing defects or degrading existing quality? (Modifiability) |       |        |          |          |        |
| degree to which the CES is composed of discrete components such that a change to one component has minimal impact on other components? (Modularity) |       |        |          |          |        |
| degree of effectiveness and efficiency with which test criteria can be established for the CES and tests can be performed to determine whether those criteria have been met? (Testability) |       |        |          |          |        |
- SECURITY is composed by the following quality attributes:
  - Authenticity
  - Confidentiality
  - Integrity
  - Non-repudiation

Would you be willing to take extra cost and effort to increase the
degree to which the CES...

* 

Please choose the appropriate response for each item:

|                                   | Never | Rarely | Possibly | Probably | Always |
|-----------------------------------|-------|--------|----------|----------|--------|
| degree to which the identity of a subject or resource of the CES can be proved to be the one claimed? (Authenticity) |   O   |   O    |   O      |   O      |   O    |
| degree to which the CES ensures that data are accessible only to those authorized to have access? (Confidentiality) |   O   |   O    |   O      |   O      |   O    |
| degree to which the CES prevents unauthorized access to, or modification of, computer programs or data? (Integrity) |   O   |   O    |   O      |   O      |   O    |
| degree to which actions or events regarding the CES can be proven to have taken place, so that the events or actions cannot be repudiated later? (Non-repudiation) |   O   |   O    |   O      |   O      |   O    |

- COMPATIBILITY is composed by the following quality attributes:
  - Co-existence
  - Interoperability

Would you be willing to take extra cost and effort to increase the
degree to which the CES...
Please choose the appropriate response for each item:

| Requirement                                                                 | Never | Rarely | Possibly | Probably | Always |
|----------------------------------------------------------------------------|-------|--------|----------|----------|--------|
| can perform its required functions efficiently while sharing a common      | ○     | ○      | ○        | ○        | ○      |
| environment and resources with other systems, without detrimental impact  |       |        |          |          |        |
| on any other system? (Co-existence)                                        |       |        |          |          |        |
| can exchange information with other systems and use the information that  | ○     | ○      | ○        | ○        | ○      |
| has been exchanged? (Interoperability)                                     |       |        |          |          |        |

- **PORTABILITY** is composed by the following quality attributes:
  - Adaptability

Would you be willing to take extra cost and effort to increase the degree to which the CES...

Please choose the appropriate response for each item:

| Requirement                                                                 | Never | Rarely | Possibly | Probably | Always |
|----------------------------------------------------------------------------|-------|--------|----------|----------|--------|
| can effectively and efficiently be adapted for different or evolving      | ○     | ○      | ○        | ○        | ○      |
| hardware, software or other operational or usage environments? (Adaptability)|       |        |          |          |        |

- **SATISFACTION** is composed by the following quality attributes:
  - Trust

Would you be willing to take extra cost and effort to increase the degree to which...

Please choose the appropriate response for each item:
a user or other stakeholder has confidence that a product or system will behave as intended? (Trust)

- FREEDOM FROM RISK is composed by the following quality attributes:
  - Economic Risk Mitigation
  - Environmental Risk Mitigation
  - Health and Safety Risk Mitigation

Would you be willing to take extra cost and effort to increase the degree to which the CES...

*Please choose the appropriate response for each item:

|                      | Never | Rarely | Possibly | Probably | Always |
|----------------------|-------|--------|----------|----------|--------|
| mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use? (Economic Risk Mitigation) |       |        |          |          |        |
| mitigates the potential risk to property or the environment in the intended contexts of use? (Environmental Risk Mitigation) |       |        |          |          |        |
| mitigates the potential risk to people in the intended contexts of use? (Health and Safety Risk Mitigation) |       |        |          |          |        |

Trade-offs

A quality attribute trade-off is a relation between two quality attributes that are incompatible and can not be individually fully optimized for a given system, which means that designers have to consider a balance between both to satisfy stated requirements. Knowing trade-off relations between quality attributes
is useful to support design decisions.

Based on the quality model below (mouseover attributes for definitions), identify at least 3 trade-off relations between two attributes that you consider challenging and very important to design CES. Consider that each relation is uni-directional: although improving quality attribute A negatively impacts quality attribute B, improving attribute B does not necessarily negatively impacts attribute A. **Obs: consider only quality attributes in grey boxes.**

Example of filling:

| Quality Attribute A | Quality Attribute B |
|---------------------|---------------------|
| Trade-off 1 Co-existence | Capacity |
| Trade-off 2 Trust | Adaptability |
| Trade-off 3 Integrity | Modularity |

Quality Model

Please fill in at least 6 answers

| Trade-off | Quality Attribute A | Quality Attribute B |
|-----------|---------------------|---------------------|
| Trade-off 01 |                     |                     |
| Trade-off 02 |                     |                     |
| Trade-off 03 |                     |                     |
| Trade-off 04 |                     |                     |
| Trade-off 05 |                     |                     |
| Trade-off 06 |                     |                     |
| Trade-off 07 |                     |                     |
| Trade-off 08 |                     |                     |
| Trade-off 09 |                     |                     |
| Trade-off 10 |                     |                     |
| Trade-off 11 |                     |                     |
| Trade-off 12 |                     |                     |
| Trade-off 13 |                     |                     |
| Trade-off 14 |                     |                     |
| Trade-off 15 |                     |                     |

Final considerations
Do you think that QM4CES will be useful to support software engineering activities to develop CES? *

Choose one of the following answers

Please choose only one of the following:

- [ ] Strongly disagree
- [ ] Disagree
- [ ] Neither agree nor disagree
- [ ] Agree
- [ ] Strongly agree

Please, select architectural design phases that can benefit from QM4CES. *

*Only answer this question if the following conditions are met:
Answer was ‘Agree’ or ‘Strongly agree’ at question ‘20 [FINAL01]’ (Do you think that QM4CES will be useful to support software engineering activities to develop CES?)

Check all that apply

Please choose all that apply:

- [ ] Architectural Analysis (architecture requirements)
- [ ] Architectural Synthesis (architecture documentation)
- [ ] Architectural Evaluation (architecture appropriateness)
- [ ] None
- [ ] Other: [ ]

Select which approaches can be supported by QM4CES. *

*Only answer this question if the following conditions are met:
Answer was ‘Agree’ or ‘Strongly agree’ at question ‘20 [FINAL01]’ (Do you think that QM4CES will be useful to support software engineering activities to develop CES?)

Check all that apply

Please choose all that apply:

- [ ] ADD (Attribute-Driven Design)
- [ ] AIW (Architecture Improvement Workshop)
- [ ] ARID (Active Reviews for Intermediate Designs)
- [ ] ATAM (Architecture Tradeoff Analysis Method)
- [ ] CBAM (Cost Benefit Analysis Method)
- [ ] DCAR (Decision-Centric Architecture Reviews)
- [ ] MTW (Mission Thread Workshop)
- [ ] PALM (Pedigreed Attribute elicitation Method)
- [ ] QAW (Quality Attribute Workshop)
- [ ] SAAM (Software Architecture Analysis Method)
- [ ] Other: [ ]
Please, explain why you do not agree (or is neutral) QM4CES can be useful to support architectural activities for CES.*

Only answer this question if the following conditions are met:
Answer was 'Strongly disagree' or 'Disagree' or 'Neither agree nor disagree' at question '20 [FINAL01]' (Do you think that QM4CES will be useful to support software engineering activities to develop CES?)

Please write your answer here:

Is there any quality attribute that you consider relevant for CES but was not present in QM4CES? You can use the model below to identify other relevant quality attributes. Quality attributes in green boxes were selected for QM4CES and you already evaluated them. The others, in red boxes, were not selected for QM4CES.*

Please choose only one of the following:
- Yes
- No

Please, indicate quality attributes that are missing. You do not need to provide definitions for quality attributes you took from the quality model above.

Only answer this question if the following conditions are met:
Answer was 'Yes' at question '24 [FINAL05]' (Is there any quality attribute that you consider relevant for CES but was not present in QM4CES? You can use the model below to identify other relevant quality attributes. Quality attributes in green boxes were selected for QM4CES and you already evaluated them. The others, in red boxes, were not selected for QM4CES.)

Please fill in at least one answer

| Quality Attribute | Definition | Comment |
|-------------------|------------|---------|
| Quality Attribute 01 |            |         |
| Quality Attribute 02 |            |         |
| Quality Attribute 03 |            |         |
| Quality Attribute 04 |            |         |
| Quality Attribute 05 |            |         |
| Quality Attribute 06 |            |         |
| Quality Attribute 07 |            |         |
For each new quality attribute suggested in the previous question, would you be willing to take extra cost and effort to increase it in a CES? *

Only answer this question if the following conditions are met:
Answer was ‘Yes’ at question ‘24 [FINAL05]’ (Is there any quality attribute that you consider relevant for CES but was not present in QM4CES? You can use the model below to identify other relevant quality attributes. Quality attributes in green boxes were selected for QM4CES and you already evaluated them. The others, in red boxes, were not selected for QM4CES.)

Please choose the appropriate response for each item:

Only answer this question for the items you selected in question FINAL06 (‘Please, indicate quality attributes that are missing. You do not need to provide definitions for quality attributes you took from the quality model above.’)

Only answer this question for the items you did not select in question FINAL06 (‘Please, indicate quality attributes that are missing. You do not need to provide definitions for quality attributes you took from the quality model above.’)

| Quality Attribute 01 | Never | Rarely | Possibly | Probably | Always |
|----------------------|-------|--------|----------|----------|--------|
| Quality Attribute 02 |       |        |          |          |        |
| Quality Attribute 03 |       |        |          |          |        |
| Quality Attribute 04 |       |        |          |          |        |
| Quality Attribute 05 |       |        |          |          |        |
| Quality Attribute 06 |       |        |          |          |        |
| Quality Attribute 07 |       |        |          |          |        |
| Quality Attribute 08 |       |        |          |          |        |
| Quality Attribute 09 |       |        |          |          |        |
| Quality Attribute 10 |       |        |          |          |        |

Indicate any quality attribute trade-off relation between the new quality attributes you are suggesting and other quality attributes.

Only answer this question if the following conditions are met:
Answer was ‘Yes’ at question ‘24 [FINAL05]’ (Is there any quality attribute that you consider relevant for CES but was not present in QM4CES? You can use the model below to identify other relevant quality attributes. Quality attributes in green boxes were selected for QM4CES and you already evaluated them. The others, in red boxes, were not selected for QM4CES.)

| Trade-off 01 | Suggested Quality Attribute | Another Quality Attribute |
|-------------|----------------------------|---------------------------|
| Trade-off 02 |                            |                           |
| Trade-off 03 |                            |                           |
| Trade-off 04 |                            |                           |
| Trade-off 05 |                            |                           |
| Trade-off 06 |                            |                           |
| Relation 07 |                            |                           |
**Suggested Quality Attribute** | **Another Quality Attribute**
--- | ---
Trade-off 08 |  
Trade-off 09 |  
Trade-off 10 |  

Feel free to leave any information you consider useful for this research

Please write your answer here:

Thanks for your participation!

Sooner you will receive a confirmation email that your responses have been saved. If you are interested in the results of my research just drop me an email so I can contact you as soon as I finish my work.

Sincerely,
Brauner Oliveira (brauner@usp.br)

Submit your survey.
Thank you for completing this survey.