Master Thesis
Software Engineering
Thesis no: MSE-2009:17
September 2009

Balancing Dependability Quality Attributes Relationships for Increased Embedded Systems Dependability

Saleh Al-Daajeh

Supervisor:
Professor Mikael Svahnberg

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 2 x 20 weeks of full time studies.

Contact Information:
Author:
Saleh Al-Daajeh
E-mail: saleh.aldajah@yahoo.ca
University advisor:
Prof. Miakel Svahnberg
School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 385 000
Fax : +46 457 271 25

Abstract

Embedded systems are used in many critical applications of our daily life. The increased complexity of embedded systems and the tightened safety regulations posed on them and the scope of the environment in which they operate are driving the need for more dependable embedded systems. Therefore, achieving a high level of dependability to embedded systems is an ultimate goal. In order to achieve this goal we are in need of understanding the interrelationships between the different dependability quality attributes and other embedded systems' quality attributes. This research study provides indicators of the relationship between the dependability quality attributes and other quality attributes for embedded systems by identifying the impact of architectural tactics as the candidate solutions to construct dependable embedded systems.

Acknowledgment

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I am deeply indebted to my supervisor Dr. Mikael Svahnberg from the Blekinge Institute of Technology whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis.

Blekinge Institute of Technology school of engineering staff, my colleagues who supported me in my research work. I want to thank them for all their help, encouragement and valuable recommendations.

Especially, I would like to give my special thanks to my brother Prof. Saud Aldajah, my parents,and my family whose trust and patient love have pushed me to the limit to complete this work.

I am also very much thankful to Prof. Claes Wohlin, whose guidance, and ideas are one of the reasons of this thesis success. This copy is specially dedicated to Professor Claes Wohlin.

Wish you all the best...

Thank you very much...

Contents

List of Figures

List of Tables

Balancing Dependability Quality Attributes Relationships for Increased Embedded Systems Dependability

Saleh Al-Daajeh
School of Engineering
Blekinge Institute of Technology
Ronneby, Sweden
Saleh.aldajah@yahoo.ca

Abstract

Embedded systems are used in many critical applications where a failure can have serious consequences. Therefore, achieving a high level of dependability is an ultimate goal. However, in order to achieve this goal we are in need of understanding the interrelationships between the different dependability quality attributes and other embedded systems' quality attributes. This research study provides indicators of the relationship between the dependability quality attributes and other quality attributes for embedded systems by identifying the impact of architectural tactics as the candidate solutions to construct dependable embedded systems.

1 Introduction

An Embedded System is a computer system designed to perform certain dedicated functions. It is usually embedded as a part of a complete device including hardware. Embedded systems pervade modern society. From consumer electronics, to automobiles, to satellites, they represent one of the largest segment of the software industry.

Since embedded systems are so pervasive, our society has come to depend on these systems for its day-to-day operation. Given the pervasiveness of embedded systems, especially in the area of highly dependable and safety-critical systems, it is necessary to consider dependability of the system in early stages its development.

Dependability is a tangible high-level attribute for a system. It consists of a small subset of different quality attributes. Dependability quality attributes and their achievement play an important role in describing the degree to which trust can be justifiably placed on a computer system. This is taken to include attributes such as availability, reliability, integrity, confidentiality, safety, and maintainability for the system.

When developing dependable embedded systems it is important to achieve sufficient levels of dependability quality attributes. Therefore, it is necessary to consider what factors affect the achievement of these quality attributes to the software such as the nature of the inter-relationship between them and other embedded systems quality attributes.

There are several studies that investigates quality attributes relationships built on different bases such as experiences, academia, and literature [1] [2]. However, all of these studies agree that the relationships between quality attributes exist and play a vital role in sustaining a sufficient level of quality to a software system.

The relationship between quality attributes which supports the achievement of other quality attributes is considered as a positive relationship. If the achievement of a certain quality attribute can hinder other quality attributes achievement, then the relationship is considered as a negative relationship. The rela-

tionship between quality attributes can also be identified as independent when having no impact on the achievement of other quality attributes.

Dependability is obviously a very desirable and important attribute of an embedded system; however, in order to achieve dependability of an embedded system, it is crucial to understand the inter-relationships between dependability quality attributes and other system quality attributes. Understanding of dependability quality attributes relationships minimize the chances of the occurrence of unwelcome system behavior during runtime and it increases the chance that the developed system will meet its specifications by selecting appropriate design decisions at early stages of the systems development [3]. Moreover, the benefits of understanding the nature of quality attributes' relationships as stated in [1], will aid in deciding which quality attributes to prioritize and which one to forsake during the development stage.

This research builds on the findings of previous studies investigating the architecture of dependable embedded systems and studies investigating quality attributes relationships. It has two main contributions:
The first one is to design a framework which is used to assess quality attributes relationships at the software architecture stage using two pieces of information; architectural tactics and quality attributes sensitive scenarios. Secondly, the application of the aforementioned framework on dependability and embedded systems quality attributes.

This article is organized as follows. Section 2 further describes the scope of this study. Section 3 presents the research aims and objectives, research questions and systematically explains the quality attributes relationship assessment framework. Section 4 provides the approach for conducting the research. Section 5 discusses the frameworks potential strengths and weaknesses. The article is concluded in section 6.

2 Background and Motivation

This study is primarily concerned with embedded systems and dependability of embedded systems, which is explained in sections 2.1 and 2.2. The software architecture plays a vital role in achieving quality attributes [4] [5] [6], which is discussed in the context of embedded systems in section 2.3. As stated in the introduction, it is always necessary to compromise between different quality attributes, which is further discussed in section 2.4. In section 2.5, the aforementioned topics are connected by discussing how to ensure that the relevant quality attributes are going to be met in an embedded systems software architecture through the use of software architecture evaluations. The section is summarized in 2.6.

2.1 Embedded Systems

Embedded systems constitute a large share of today's products. When developing embedded system software, quality is a key characteristic, on its own or its implications. Managing software quality is necessary to deliver embedded system software functioning in a useful, safe and reliable way [7].

Embedded systems are recognized as computer systems that are part of a larger system and perform specific functionalities under certain constraints such as a computer system used in an aircraft or rapid transit system. The *IEEE Standard Glossary of Software Engineering Terminologies* defines embedded systems software as a part of a larger system which performs some of the requirements of that system [8].
Due to embedded systems operational environment characteristics and common requirements they are known as safety-critical systems and hard-real-time systems [9] [10].

According to Crnkovic, the study of embedded systems shows that the various types of embedded systems share common requirements such as: dependability, real-time constraints, resource consumption and life-cycle properties [11]. Embedded systems are expected to be failure-free. This requirement hinders embedded systems to deliver a justifiable service at a reliance level given a potential disturbance to its services, for example, failure in a component or a mishap in using the system. Furthermore, embedded systems' operations have specific deadlines, milestones, etc.

Various types of embedded systems have specific

execution time describing the duration of their operations and delivering output. Thus, embedded systems shall not miss the pre-defined milestones in processing and delivering output of their operations. In most cases, embedded systems are built with limited resources, for example, limited memory resources. Most embedded systems are required to successfully operate and deliver services using few hardware and software resources (e.g. less power consumption and few lines of code "SLOC").

Additionally, embedded systems are expected to have long life times. Generally, embedded systems are developed to deliver service for long periods of time such as embedded systems in satellites where the life cycle is estimated to be from 3-10 years.

In most cases, the development of embedded systems software faces conflicts in the requirements placed on them (e.g. high availability requirements vs. limited memory resources). This conflict is due to the tightened safety regulations, the scope of the environment in which this type of systems operates, and limited resources [9][12]. When developing embedded systems software, it's important to achieve a sufficient level of quality and precision for the product and the development processes [13] [14].

Embedded systems development has a growing demand for more dependable software, especially as the dependability of embedded systems software influences important quality attributes of the systems (e.g. availability, reliability, safety, etc.) [15]. There are different approaches to be undertaken at different stages of the software development process to support the implementation of dependability in embedded systems software under different considerations [9] [14] and [16].

Unfortunately, in most cases the implementation of dependability in embedded systems is left until later stages in the software development, which may cause some complications, and difficulties in reintroducing dependability to the system [17]. Furthermore, it will result in increased software development costs [4]. However, the implementation of dependability in embedded systems software at earlier stages of its development life cycle (e.g. Software Architecture) requires an understanding of the nature of the interrelationship between dependability quality attributes and other desired quality attributes of the embedded system software [14].

Embedded systems software and computer-based software systems are considered to be alike from the quality perspective [18]. Embedded systems quality attributes are deliberated the same way as any software quality attributes.

To the knowledge of the investigator, there are no studies investigating embedded systems quality attribute except the one conducted by Sherman [18]. In his study, general quality attributes for different embedded systems are introduced. The quality attributes types for embedded systems included in this study are hardware quality attributes, software quality attributes, and quality attributes that are related to both embedded systems software and hardware components.

2.2 Embedded Systems Dependability

Embedded systems dependability is a very important concern in embedded systems [10] [11]. For various types of systems, dependability is a small subset of quality attributes. According to [Avizienis et al.] and [Laprie et al.], dependability is defined as:

"The property of a system that delivers justifiably services at a reliance level and the ability of the system to avoid failures that are serious and numerous" [19] [20].

Dependability quality attributes are defined with respect to what the acceptable behavior is of the system in case of faults, attacks, mishaps and failures that occur when the system is operating, and what the acceptable amount is of effort required for the modifications needed to correct errors. For example, the system can be safe but not dependable by delivering incorrect, late or even early output in some cases. According to the software architecture society, dependability quality attributes can be classified as runtime and non-runtime quality attributes [20]. Moreover, more than one dependability quality attribute can share the same focus to achieve the overall dependability concerns.

The followings are the dependability quality attributes definitions and classification according to [M. Barbacci et al.] and [J. Laprie et al.] [15] [21]:

Runtime Dependability Quality Attributes:

- Availability: readiness of service for authorized users. It is measured by the duration it would take an intruder to cause a denial of service. This quality attribute focuses on the system behavior versus the faults encountered during the system operation.

- Reliability: continuity of service. The system is expected to perform its task in spite of the existence of some faults. This quality attribute is also concerned with demonstrating acceptable behavior of the system when faults are encountered.

- Integrity: non-occurrence of improper alternation of information. This quality attribute is concerned with the system behavior in case of attacks.

- Confidentiality: non-occurrence of unauthorized disclosure of information as system data and programs are resistant to unauthorized modifications. This quality attribute describes how the system will behave in case of attacks.

- Safety: non-occurrence of catastrophic consequences for the user(s) and in the operation environment. This quality attribute describes how the system should deal with mishaps and/or failures when they occur.

Non Run-Time Dependability Quality Attribute

- Maintainability: aptitude to undergo repairs and evolution. This quality attribute demonstrates the ease of modifying the software or system component to correct faults.

2.3 Engineering Quality of Embedded Systems via Software Architecture

Usually the second stage of the software development life cycle is the software architecture or a high level design of the system. Software architecture supports the achievement of quality for a software system by providing design decisions and specifications that satisfies the means of quality attributes [4] [5] [6].

[C. Ebert et al.] denoted in their study of monitoring quality achievement progress throughout the software development processes that the quality achievement reaches 45% of the total quality before the software exits the virtual architecture quality gate (SAG) [6].

The Software Architecture Society has been active in this field and it has shown how software architecture constrains the achievement of quality [5] [22] [23] [24] [25] [26]. According to [L. Bass et al.], software architecture is the structure or structures of the system, which compromise software components, the externally visible properties of those components and the relationships between them [5]. Software architecture is documented via multiple methodologies and structured via different styles. Thus the architecture of embedded systems software provides the foundation on which system developers can achieve quality throughout the system development processes and evaluate a system's design in respect to the desired quality attributes [5] [22] [17]. Most importantly, the achievement of dependability quality attributes in the software development process is strongly related to the software architecture.

The achievement of quality attributes for a system in software architecture can be separated into two levels: requirements and solutions. The software architecture community has developed several frameworks used to elicit, specify and categorize quality requirements [5] [27] [28]. These frameworks are used to create what is known as quality attributes general scenarios, which help in developing quality attributes concrete scenarios and therefore assist in evaluating software architectures [5].

General scenarios can be used as a template to define concrete scenarios for quality attributes

scenarios. Quality attributes concrete scenarios are methods to create a description of the problem (constraint or requirement) which questions whether the software architecture candidate solutions satisfies the desired non-functional requirements set on the system or not [16].

The solution for achieving software quality attributes during the software architecture stage is supported by selecting appropriate architectural strategies or "tactics" [5]. Tactics are known as design decisions that influence and control the response of quality attributes [5] [25]. [L. Bass et al.] and [W. Wu et al.], suggest multiple tactics to support the achievement of different dependability quality attributes to a system [5] [29]. However, most of these tactics are conducted based on experience rather than theory according to [Bachmann et al.][30].

The software architecture in terms of connectors and components provide a starting point for revealing the areas of potential change which might affect the system dependability. The overall achievement of dependability is dependent on the individual achievement of its subset of quality attributes. According to M. Larsson, the software's dependability quality attributes achievement is strongly related to the software architecture stage of the software development life cycle [10]. Moreover, architectural level reasoning of dependability is an important theme to consider while developing dependable embedded systems software.

2.4 Trade-Offs in Embedded Systems Software and Dependability

Trade-offs is made subconsciously on a daily basis. In embedded systems software development, system quality attributes are among the important subjects that in some situations require trade-offs especially when embedded systems face conflicts in requirements. Trade-offs differs from one situation to another, and a rigorous analysis might be required. Any technique that helps to make or consolidate the trade-off process or decision can be considered as a trade-off technique. Trade-offs can be done via multiple techniques in the same process to evaluate the results. P. Berander et al. provides a classification of the trade-offs techniques that can be demonstrated into the following three categories [24]:

- Experienced based: which depends mainly on the experience for supplying the needed information to performing the trade-off.
- Model based: which depends on constructing e.g. a graphical model for illustrating the relations between trade-offs entities, thus facilitating the trade-off.
- Mathematically based: which relies on mathematical formulas for constructing and representing the trade-off, thus making it possible to feed the mathematical construct with appropriate values and receiving the best solutions. For example, Analytical Hierarchy Process "AHP", which helps to control the trade-offs and can help to conduct the right selection depending on the factors that influence the model [31].

The critical part of trade-offs processes is to define the factors that have an impact on the selection of solution(s). However, trade-offs can be made at different levels and can also be constructed of several sub-trade-offs within one level which makes trade-offs dependent on the depth and strength of analysis required.

Software architects are the ones who make the final decisions on how the system shall meet its specifications and customers expectations. In embedded systems software architecture trade-offs are done at two main levels with respect to software dependability and other quality constraints. The first level is to prioritize the desired system quality attributes, for example, a landing system must have high availability while the system is not required to be reliable for longer periods.

The second level of trade-offs is the selection of a software architecture pattern/style that supports the pre-prioritized quality attributes the best way possible [24]. However, in the second main trade-offs a sub-trade-offs is needed in the form of technical reviews. The intention of technical reviews is to evaluate the solution(s) against the predefined specifica-

tions and compliance with standards and other documentations [31]. In this sub-trade-off software architects are required to select and calibrate the best candidate tactics that support dependability quality attributes and have least impact on other quality attributes of the same system software.

To make the appropriate selection, there are many factors to consider, for example if the selected architecture tactic supports the achievement of one quality attribute or more and if this tactic has a less negative impact on other system quality attributes and could therefore fulfill the constraints made on the system with few undesired or expected consequences. However, software architects must consider the nature of the interrelationships between dependability quality attributes and other quality attributes to select and calibrate the best representative architectural strategies and therefore create an architecture that meets its predefined specifications.

2.5 Software Architecture Evaluation

To achieve quality for embedded system software at the software architecture stage, it is important to evaluate the system architecture with respect to the desired quality and the requirements placed on the system. In general, software architecture evaluation aims at providing evidence as to whether the architecture is suitable with respect to its functional and nonfunctional requirements.

Software architecture evaluation methods are usually based on critical analysis of how the system fulfills its specification. There are several methods that can be used to assess software architecture and predict the quality of software at the architectural level (e.g. ATAM, SAAM, etc.) [5].

There are two general categories suggested for software architecture evaluation methods according to [R. Kazman et al.]: qualitative analysis and quantitative measurement. The qualitative analysis of software architecture is usually based on questionnaires, checklists, and scenarios. While quantitative measurements consist of simulations and metrics, for example modeling and testing. Quantitative measurement aims at estimating the fulfillment of a quality attribute in terms of probabilities [14] [28].

Recently, there are several studies conducted to evaluate architectural styles/patterns that support different quality attributes [5] [32] [33] [34] [35]. Hence the impact assessment of using an architectural tactic is steered by evaluators logical reasoning using their experiences. In this research we adopt objective reasoning as the method to evaluate the impact of using proposed architectural tactics on dependability quality attributes inter-relationships and their relationship with other embedded systems quality attributes. However, further recommendations on software architecture evaluation methods can be found in [24] [32] [33] [34].

2.5.1 Objective Reasoning

Software architecture evaluation does not only help the evaluators to be able to reason about the achievement of software quality attributes and to have a certain level of predictability of the system's behavior, but it also helps to select suitable tactics to achieve the desired quality attributes and hence supports the trade-offs and therefore the selection of an appropriate architectural pattern.

Objective reasoning is one of the approaches used to assess the software quality requirements achievement in software architecture through reasoning based on logical arguments [13] [25]. According to [L. Zhu et al.], the analysis of software architecture tactics with respect to the desired quality attribute might not be sufficient [22]. He suggests that evaluating software architectural tactics using techniques such as impact analysis or walk-through might not be reliably sufficient. Instead, he suggests developing appropriate quality attributes reasoning frameworks with the help of scenario-based architectural evaluation methods to be combined with some other techniques and therefore include all the parameters which might influence the achievement of system quality attributes at the software architecture level.

2.6 Summary

Dependability is a very important concern for embedded systems. In most cases, embedded systems are required to be dependable from attacks, faults,

failures and mishaps when they are released to service. Dependability is a tangible and a high-level quality attribute that consists of six main quality attributes. Dependability quality attributes can be witnessed when the system is running or deactivated.

Usually quality attributes can be found as non-functional requirements. Hence requirements are seldom elicited and documented in a disciplined way. Software quality attributes are specified with the help of general and concrete scenarios.

There are three methods that can be adopted to achieve quality to a software system [23]. One of these three methods is engineering quality to software systems. This method aims to manifest quality attributes requirements at early stages of the system development such as software architecture. Hence this research is concerns with the implementation of dependability requirements at early system development stages. We focus on software architecture as it is one of software development initial stages were engineering quality to software system have great influence in achieving quality to software system. Additionally, software architecture plays a vital role and constrains the quality achievement to software system by providing multiple architectural tactics that is capable of satisfying the software quality attributes to some extent.

The use of architectural tactics to achieve certain dependability quality attribute affect the achievement of other quality attributes and therefore creates a relationship that could support or hinder the achievement of other quality attributes or have no impact.

In order to create a dependable embedded system possessing all the desired quality requirements at system development early stages, we need to characterize the nature of quality attributes relationships at the software architecture level and be able to provide the appropriate architectural tactics and trade-offs. In addition, we are required to understand what is the impact of using certain tactic to achieve a quality attribute on the relationship with other quality attributes.

3 Research Design

Research main objectives and underlying questions are shown in section 3.1 and section 3.2 respectively. In section 3.3, the assessment of quality attributes relationships based on dependability quality attributes candidate architectural tactics impact on other quality attributes achievement evaluation with the help of quality attributes scenarios is shown as quality attributes relationships reasoning framework which also uses objective reasoning as the impact evaluation method for the proposed tactics.

3.1 Research Aims and Objectives

This research aims at understanding the interrelationship between dependability quality attributes and other embedded systems quality attributes. The research objective is to increase the likelihood of achieving dependability in embedded systems by utilizing the best trade-offs between candidate architectural tactics used to achieve dependability for embedded systems. In order to achieve the research goals, the impact of the proposed architectural tactics on the dependability quality attributes and the other quality attributes of an embedded systems are studied.

3.2 Research Questions

The research goals can be achieved by answering the following five questions:

1. What are the high level embedded systems quality attributes?

2. What are the architectural strategies “tactics” used to achieve dependability quality attributes?

3. When utilizing a certain tactic, what is the nature of the impact on other quality attribute(s)?

4. What is the best trade-offs of candidate architectural tactics used to achieve certain dependability quality attribute that influence other dependability quality attributes?

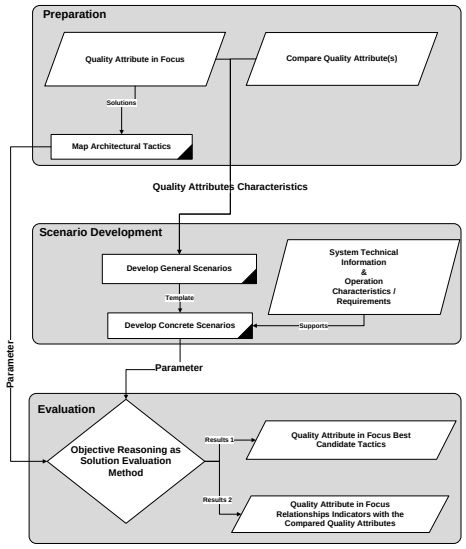

Figure 1: Quality Attributes Relationship Reasoning Framework

5. What are the interrelationship between dependability quality attributes and other embedded systems quality attributes nature and characteristics?

3.3 Quality Attributes Relationship Reasoning Framework

This section describes the steps of constructing the framework used to reveal the quality attributes relationships.

The design of this framework is triggered by the model provided by [L. Zhu et al.] which describes the skeleton of the relationship between patterns, tactics, evaluation and pattern validation. The proposed framework consists of three main steps: preparation, scenario development and evaluation [22]. The following sections explain these main steps which are also shown in Figure 1.

3.3.1 Preparation

The preparation objective is to bring relationship main parts "quality attributes" to the investigation process. Moreover, this step is designed to produce two outputs; first outcome is quality attributes characteristics of both relationship parts which will be an input for the second step in this framework. Second outcome is candidate architectural tactics which can be used to achieve quality attributes as a input parameter for the third step of this framework. However, this step consists of two main sub-steps:

- *A. Identify Quality Attributes in Focus and Compared Quality Attributes.*

This step focuses on identifying "Quality Attributes in Focus" as the main part of the study to investigate its relationship with the "Compared Quality Attributes" as second main part of the investigated relationship. This step aims to embrace "Quality attributes in Focus" concerns and characteristics to support the process of selecting appropriate architectural tactics. In addition, this step deliver both "Quality Attributes in Focus" and " Compared Quality Attributes" characteristics to enable generating quality attributes scenarios.

- *B. Map Architectural Tactics to the Dependability Quality attributes as Candidate Solutions.*

Selecting and calibrating appropriate architectural tactics that can be used to achieve "Quality Attributes in Focus" is the aim of this sub-step. This sub-step requires understanding of the quality attributes in focus and its sub-characteristics "quality criteria". This step is designed to deliver "Quality Attributes in Focus" candidate architectural tactics an independent parameter for the evaluation process in the third step of this framework "Evaluation-Step".

3.3.2 Scenario Development

The objective of this step is to assess the characteristics of the quality attributes by using quality attributes general and concrete scenarios. This step receives both "Quality Attribute in Focus" and

"Compared Quality Attributes" as input. There are several matrices used to present quality attributes that are discussed in [22]. In this research, we consider the matrix suggested by [5]. This matrix consist of six elements, each element represents an important part describing the quality attribute. Table 1 depicts these elements of this matrix. Scenario Development step consists of two main sub-steps:

- *A. Develop Quality Attribute General Scenarios.*

As stated before, general scenarios are used to explore the important elements of the quality requirements. This sub-step is needed to provide a template by which concrete scenarios are built upon. General Scenarios are distilled from the common requirements placed on "Quality Attribute in Focus" and "Compared Quality Attributes".

Table 1: General Scenarios Matrix According to [L. Bass et al.] [19]

Elements	Description
Source of Stimulus	An entity (human, computer, etc..) that generate the stimulus.
Stimulus	A condition that need to be considered when it arrives at a system.
Simulated artifacts	Some artifacts that are simulated (e.g. whole system, parts of the system).
Environment	A system's condition when a stimulus occurs.
Response	The activity undertaken after stimulus arrival.
Response Measure	The response to the stimulus should be measurable in some fashion so that the requirement can be tested.

- *B. Develop Quality Attribute Concrete Scenarios.*

The final sub-step is to develop a concrete scenario for "Quality Attributes in Focus" and "Compared Quality Attributes". The main objective of this sub-step is to fully address the studied quality attributes. However, quality attributes concrete scenarios can be supported by additional data such as system technical information. This sub-step forwards the second dependent parameter needed for the "Evaluation-Step".

3.3.3 Evaluation

In this step, we used the aforementioned method "objective reasoning" to evaluate the impact of candidate architectural tactics on "Quality Attributes in Focus".

The evaluation method has two main parameters that are produced in the preparation and Scenario Development steps. In addition, different factors that may influence the assessment of the architectural tactic impact are considered such as : design rational, domain context, and implementation factors.

4 Execution

In this research, the main focus is on revealing the nature of the inter-relationships between dependability quality attributes and their relations with other embedded systems quality attributes. Research results are conducted by following a systematic approach as presented in section 3. Some modifications to the generic framework are necessary to instantiate it for dependability and embedded system quality attributes. These modifications are shown in Figure 2.

4.1 Preparation

This step identifies dependability quality attributes and embedded systems at a high level of detail. It consists of three main sub-steps:

- *A. Identify Dependability Quality Attributes and Embedded Systems Quality Attributes*

This research is focused on discovering the inter-relationships between dependability different quality attributes and their relationship with other embedded systems quality attributes. This sub-step adopt dependability quality attributes respectively as the "Quality Attributes in Focus" and receives other embedded systems quality attributes as "Compared Quality Attributes ".

- *B. Map Architectural Tactics to Dependability Quality Attributes*

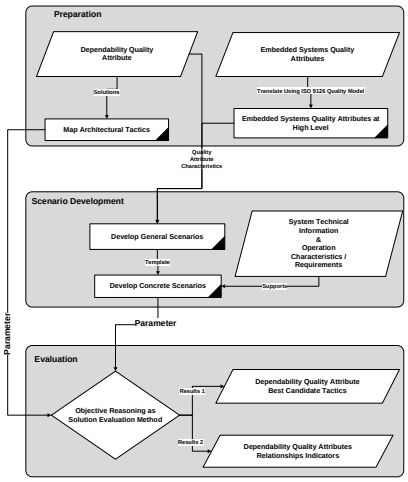

Figure 2: Dependability and Embedded Systems Quality Attributes Relationships Reasoning Framework

As stated, this step is dedicated to identify architectural tactics that are sufficient to satisfy dependability quality attributes concerns and characteristics. According to [N. Lassing et al.], the dependability quality attributes availability and reliability share the same architectural tactics [27]. From a dependability perspective, the quality attributes availability and reliability have faults as a common concern. The dependability quality attributes integrity and confidentiality are concerned with developing dependable systems to secure a system from attacks. Engineering security to software system consider both integrity and confidentiality to share the same concerns. Therefore, tactics that are used to achieve the quality attribute "Security" can be undertaken to achieve dependability quality attributes integrity and confidentiality.

Safety describes how the system shall be dependent from mishaps and failures. This quality attribute has two concerns from a dependability perspective: interaction complexity and coupling strength.
Interaction complexity describes how the system is supposed to be safe from unfamiliar, unplanned, expected, and either not visible or not immediately lucid sequences [22]. Coupling strength describes the process dilation and response manner. Safety architectural tactics are greatly influenced by the experience of the software architect, the context in which the system is expected to operate and interact with other parties, and the testability of the system.

According to the ISO 9126 quality model the dependability quality attribute maintainability relates to the ease and effort needed to modify the software or system component to correct faults [24]. This quality attribute is dependent on four sub-characteristics according to ISO 9126 quality model. Taking into account dependability concerns, maintainability sub-characteristics can be defined as the following:

- Analyzability : Attributes that describes the effort required for the diagnosis of deficiencies or cases of failures.
- Changeability: Attributes relate to the effort needed for fault removal.
- Stability: Attributes that is related to the risk of the unexpected effect of modifications.
- Testability: Attributes that relate to the effort needed for validating the modified software.

Decomposing the maintainability quality attribute to its sub-characteristics helps to identify candidate tactics. Modifiability and testability tactics are selected to satisfy maintainability concerns. Table 2 depicts dependability quality attributes candidate architectural tactics.

- *C. Translating Embedded Systems Quality Attributes to ISO 9126*

As stated before, there is no study that shows embedded systems quality attributes but Sherman's. In his study, he shows quality attributes for embedded systems extracted out of 12 different architectures from different embedded systems products. The quality attributes shown in this study are low -level 'quality criteria'. These quality criteria can be interpreted

Table 2: Dependability Quality Attributes Candidate Architectural Tactics

Availability& Reliability	Integrity & Confidentiality	Safety	Maintainability
Fault Detection:	***Resisting Attack:***	***Failure Detection:***	***Manage Input/Output:***
Ping/Echo	Authenticate Users	Timeout	Record/Playback
Heartbeat	Authorize Users	Time Strap	Separate Interfaces From Implementation
Exception	Maintain Data Confidentiality	Sanity Checking	Specialized Access Routines /Interfaces
Recovery Preparation and Repair:	Maintain Integrity	***Failure Containment:***	
Voting	Limit Exposure	Redundancy	***Localize Changes :***
Active Redundancy	Limit Access	Replication	Semantic Coherence
Passive Redundancy	***Detecting Attacks:***	Functional Redundancy	Anticipate/ Expected Changes
Spare	Intrusion Detection	Analytical Redundancy	Generalize Module
Recovery Reintroduction:	***Recovering From an Attack:***	***Recovery:***	Limit Possible Options
Shadow	Restoration	Fix the Error	Abstract Common Services
State Resynchronization	Identification by Audit Trail	Rollback	***Prevention of Ripple Effect :***
Rollback		Degradation	Hide Information
Prevention :		Configuration	Maintain Existing Interface
Removal From Service		***Masking:***	Restrict Communication Paths
Transactions		Voting	Use an Intermediary
Process Monitor			***Internal Monitoring :***
			Built in Monitors
			Defer Binding Time:
			Runtime Registration
			Configuration Files
			Polymorphism
			Component Replacement
			Adherence to Define Protocols

and related to high-level quality attribute differently depending on the used quality model. Moreover, the achievement of quality attributes is dependent on the achievement of its multiple quality criteria. Therefore, the findings of Sherman's study is required to be translated to a higher level using a quality model and construct a better understanding of embedded systems quality attributes characteristics and influence the chances that the system development will meet its specifications and reduce the conflicts between embedded systems quality attributes characteristics.

We have selected the ISO 9126 model to represent embedded systems quality attributes at a higher level after the comparison between different quality models such as ISO 9126, McCall, Boehm and others. The quality model ISO 9126 calibrate six areas of software quality importance (e.g. functionality, efficiency , reliability, usability, maintainability, and portability). This quality model is a fixed quality model for any type of software [24] and globally used.

In order to assess the adequacy of the translation process a survey was distributed among 15 software engineer master student to collaborate in giving the right translation of low-level embedded systems quality attributes. Table 3 depicts the embedded systems quality attributes found in [18] translated to a higher-level using the ISO 9126 quality model.

4.2 Scenario Development

This step receives the embedded systems and dependability quality attributes characteristics and translates them as quality requirements for an embedded system. This step initially characterize quality attribute important elements by using quality attribute general scenarios. General scenarios provide templates to develop concrete description of quality attributes potential requirements for a specific embedded system. However, concrete scenarios can be supported additional data such as technical information of the desired system.

In this research, we have employed the technical information provided by the Swedish Space Corporation "SSC" about the satellite "SMART 1- 2005") to support the process of generating dependability and embedded systems quality attributes concrete

Table 3: Embedded Systems Quality Attributes Translation to the ISO 9126 Quality Model

Embedded systems Quality Attributes Found in T. Sherman Study [18]	Quality Attribute Type	Translation to ISO 9126	Level of Agreement
Code size	Software Quality	Efficiency	100%
Memory usage	Software Quality	Efficiency	100%
CPU Time Used	Software Quality	Efficiency	93%
Accuracy	Software & Hardware Quality	Functionality	100%
Compatibility	Software & Hardware Quality	Portability	93%
Complexity, CPU, System, Peripherals etc.	Software & Hardware Quality	Efficiency	100%
Cost/Schedule	Software & Hardware Quality	No Equivalence	80%
Ease of Integration	Software & Hardware Quality	Portability	87%
Ease of Use/Usability	Software & Hardware Quality	Usability	100%
Expansion Capability	Software & Hardware Quality	Maintainability / Functionality	73%
Functionality	Software & Hardware Quality	Functionality	100%
System Resources Usage	Software & Hardware Quality	Efficiency	100%
Versatility/ Flexibility	Software & Hardware Quality	Maintainability	80%
Performance	Software & Hardware Quality	Efficiency	87%
Availability	Software & Hardware Quality	No Equivalence	100%
Reliability	Software & Hardware Quality	Reliability	100%
Safety	Software & Hardware Quality	No Equivalence	100%
Security	Software & Hardware Quality	Functionality	100%
Maintenance	Software & Hardware Quality	Maintainability	100%
Durability	Software & Hardware Quality	Reliability	87%
Physical Size	Hardware Quality	Efficiency	100%
Power Consumption	Hardware Quality	Efficiency	100%

scenarios [36]. The following are examples which describe the dependability quality attribute "Availability"'s general and concrete scenarios.

- Availability (General Scenario)

A satellite is expected to change its orbit (stimulus) based on the navigation commands from the control base (source of stimulus). Therefore, the navigation systems provide navigation services as close to 100% as possible (response measure) whereas planned or unplanned maintenance (stimulus) throughout its operation time (environment) shall not bring the navigation system or satellite system (artifact) services down (response measure).

Table 4 shows the dependability quality attribute (availability) general scenario using the proposed matrix as in Section 4.2.

Elements	**Description**
Source of Stimulus	Satellite Control-Base
Stimulus	Periodic Coordination Updates
Simulated artifacts	Satellite Software System
Environment	Normal Operation Conditions
Response	Up-to-Date Time Navigation services
Response Measure	Service Availability as close to 100% of satellite operation time

Table 4: Availability General Scenario

- Availability (Concrete Scenario)

SMART 1-2005 will be traveling to and from and orbit the moon (Environment). The satellite (Artifact) navigates itself using the stored coordination data. In case of an emergency (Stimulus) during its operation the satellite is directed by control-base (Source of Stimulus). The satellite is required to send and receive coordination data and change its orbit coordination accordingly (Response). Sending and receiving coordination data shall be 24 hours/day to control-base radar (Response Measure).

Table 5 shows a concrete scenario example used to assess dependability quality attribute (availability) requirements using SMART 1-2005 technical information. In order to reveal the nature of the relationships we need to assess the dependability different quality attributes and embedded systems quality attributes with both general and concrete

Elements	Description
Source of Stimulus	Satellite Control-Base
Stimulus	Emergency Case
Simulated artifacts	Satellite Navigation System
Environment	Operation Conditions
Response	Up-to-Date Time Navigation services
Response Measure	Navigation services availability measured as 24h/day

Table 5: Availability Concrete Scenario

scenarios. Dependability and embedded systems quality attributes general and concrete scenarios used in this research can be found in Appendix A.

4.3 Evaluation

4.3.1 Context and Preparation

Since the proposed tactics are built on experience and the appropriate methodology to evaluate is objective reasoning we have constructed a research survey in addition to the researchers'own evaluation. The survey design is split into three parts:

- Research Survey Participants: For this research, candidate participants are from academia and include: experienced participants in software design, experienced embedded systems software developers, and software security specialists and experts.
- Research Survey Question: we have used a question that is iterated for each architectural tactic. Moreover, we have supported the participants with an example of an answered question explaining a suitable method to provide the evaluation of the impact of using potential architectural tactic for a given dependability quality attribute. Additionally, research participants have assessed on a scale which describes the architectural tactics impact level on each quality attribute's relationships. The scale grades the level of impact from zero to four, where zero indicates no impact, one indicates a slight Impact, two refers to a minor impact, three refers to a major Impact, and four indicates a great or severe impact on the relationship quality attributes.
- Research Survey Distribution and Data Collection: Since there are a large number of candidate architectural tactics to achieve different dependability quality attributes, we have split the research survey into four main parts according to dependability different quality attributes candidate architectural tactics and participants profiles. In this research, the method used for participants sampling is the stratified method as it reduces the sampling errors [37]. Thus, we have dedicated dependability quality attributes availability and reliability candidate tactics to embedded systems software developers, academia, and to experienced software design participants. Dependability quality attributes integrity and confidentiality candidate architectural tactics were distributed among software security engineering specialists. Moreover, dependability quality attributes safety and maintainability candidate architectural tactics were distributed among participants of software engineering background. Table 6 depicts dependability quality attributes candidate architectural tactics impact evaluation survey distribution details.

4.3.2 Execution

The research survey was a take home survey. Participants were given sufficient time to answer the survey questions and be able to use the supplementary data to reasonably answer the research questions. The approach to collect research answered survey was done via email. However, data collected from the survey represents a measured impact of using a given tactic to satisfy the achievement of a certain dependability quality attribute on other dependability and embedded systems quality attributes. Moreover, we have requested that the participants justify their answers.

4.3.3 Results

The representation of the collected data shows the impact level of utilized tactic of a certain dependability quality attribute and the level of agreement be-

Participants Role	No. of Participants	Assessed Quality Attribute
Academia	2	Availability and Reliability
Software Engineering	10	Maintainability, and Safety
Software Design	2	Availability and Reliability
Security Engineering	6	Integrity and Confidentiality
Embedded System Development	2	Availability and Reliability

Table 6: Research Survey Distribution Data

tween participants' answers of the same survey. Relationships between dependability quality attributes and other embedded systems quality attributes is conducted on the base of quality attributes architectural tactics level of impact. However, tables 7, 8, 9 and 10 shows the evaluation results.

The conducted data in the aforementioned tables shows that to satisfy a certain concern of certain dependability quality attribute there are several tactics with various level of impact on other dependability and embedded systems quality attributes achievement. For example, in table 7, the level of impact when adopting "Exception" as a candidate architectural tactic to satisfy the concern "fault detection" for the dependability quality attributes availability and reliability, this tactic will greatly support the achievement of dependability quality attributes integrity and confidentiality and therefore creates a positive or supportive relationship between dependability quality attributes "Availability and Reliability" with "Integrity and Confidentiality". Utilizing the architectural tactic "Ping/Echo" to satisfy the same concern for the same dependability quality attributes is severely hindering the achievement of dependability quality attributes integrity and confidentiality to embedded systems software.
Since we have used an ordinal scale to evaluate the impact level, the assessment of the relationships impact on achieving dependability and other quality attributes is measured using the median value of impact that the proposed architectural tactics have on other dependability and embedded systems quality attributes. For example, in table 8, the relationship between dependability quality attributes "integrity and confidentiality" with embedded systems functionality quality attribute is considered a supportive relationship. As the impact median value is estimated to be at level (3) the achievement of dependability quality attributes availability and reliability have a major support in achieving functionality to embedded systems software. On the other hand, achieving dependability quality attributes " integrity and confidentiality" to embedded systems hinders the achievement of its quality attribute "portability" to a major extent. Moreover, we have used (+)sign to indicate a positive relationship,(-) sign for negative relationship.

4.4 Different Views on Quality Attributes Relationships

As stated before in section 3.1, we are also interested in knowing the nature of the relationships between quality attributes. In order to accomplish this objective we have performed qualitative analysis of other studies investigated the relationship between quality attributes results and research results. Recent studies investigated the relationships between quality attributes are based on different approaches, for example [Henningsson et al.] and [Zulaziz et al.] are based on industrial experiences [38] [39].

One additional recent study in this field of [M. Svahnberg et al.], combined different views on the relationship between quality attributes such as academia, industry, and literature and concluded with the match between different views on quality attributes relationships [1]. Table 11 provides a list of recent studies results in investigating quality attributes relationships [1] [39].

Table 7: Availability and Reliability Candidate Architectural Impact Evaluation

Dependability Quality Attributes (Availability & Reliability) | Impact Evaluation on other Dependability & Embedded Systems Quality Attributes									
Candidate Architectural Tactics	Functionality	Efficiency	Usability	Portability	Integrity	Confidentiality	Maintainability	Safety	Level of Agreement
Fault Detection									
Ping/Echo	+4	+2	0	+1	-3	-3	+4	-3	67%
Heartbeat	+4	+1	0	-1	+2	+2	+2	0	83%
Exception	+4	-1	+1	-1	+4	+4	+2	+1	83%
Recovery Preparation and Repair									
Voting	-2	-3	0	0	0	+1	-1	+1	83%
Active Redundancy	+2	-3	0	-1	-1	-3	-2	-4	83%
Passive Redundancy	+4	+2	0	+2	+2	+2	+2	+2	83%
Spare	+4	-1	-1	0	-1	+2	-1	+1	100%
Recovery Re-Introduction									
Shadow	+4	-1	0	0	-2	+1	-1	+1	67%
State Resynchronization	+4	-1	0	0	-2	+1	-1	+1	17%
Rollback	+2	-1	-1	0	-4	+1	-3	+2	83%
Prevention									
Removal from Service	+4	-1	-2	0	-2	-2	+1	+4	100%
Transaction	+4	0	0	0	+4	+2	0	+4	67%
Process Monitor	-2	-2	0	0	-2	-2	0	-2	100%
Relationship	P	N	N	N	N	P	P	P	
Impact Median Value	+1	-1	-0.5	+0.5	0	+0.5	+0.5	0	

Table 8: Integrity and Confidentiality Candidate Architectural Impact Evaluation

Dependability Quality Attributes (Integrity & Confidentiality) | Impact Evaluation on other Dependability & Embedded Systems Quality Attributes									
Candidate Architectural Tactics	Functionality	Efficiency	Usability	Portability	Availability	Reliability	Maintainability	Safety	Level of Agreement
Resisting Attacks									
Authenticate Users	+2	-3	+2	-4	0	+2	-3	-3	100%
Authorize Users	+2	-4	0	0	0	0	-4	0	100%
Maintain Data Confidentiality	+2	-1	0	0	0	0	-4	+2	83%
Maintain Integrity	+4	-4	0	0	0	+4	-3	+4	67%
Limit Exposure	0	+4	0	0	-3	+4	-4	0	67%
Limit Access	0	0	-4	0	-3	+4	0	0	67%
Detecting Attack									
Intrusion Detection	0	-3	0	-3	0	+2	-3	0	83%
Recovering From Attack									
Restoration	+3	-4	0	0	+2	+4	+1	+2	33%
Identification by Audit Trial	0	-4	0	-4	0	+4	-3	0	67%
Relationship	P	N	N	N	N	P	N	P	
Impact Median Value	+3	-2.5	-1	-3.5	-0.5	+3	-1.5	+0.5	

Table 9: Safety Candidate Architectural Impact Evaluation

Dependability Quality Attribute (Safety) | Impact Evaluation on Other Dependability & Embedded Systems Quality Attributes										
Candidate Architectural Tactics	Functionality	Efficiency	Usability	Portability	Availability	Reliability	Integrity	Confidentiality	Maintainability	Level of Agreement
Failure Detection										
Time out	0	+4	+2	-1	+2	0	0	0	0	60%
Time Strap	0	-1	0	+4	-1	+4	0	0	+4	60%
Sanity Checking	0	-1	0	+4	-4	+4	0	0	+4	80%
Failure Containment										
Redundancy	+4	-2	0	+2	+3	+4	-2	-1	+4	80%
Replication	+4	-2	0	+2	+2	+2	-1	-1	+4	80%
Functional Redundancy	+2	-2	0	+2	+2	+2	-2	-2	+2	60%
Analytical Redundancy	0	-2	0	0	0	0	0	0	0	80%
Recovery Re-Introduction										
Fix the Error	+2	-2	0	0	-2	+2	0	0	+2	80%
Rollback	0	-4	0	0	-1	+3	+4	0	+2	60%
Degradation	-4	0	0	0	-3	-3	0	0	0	80%
Reconfiguration	+4	-1	0	+1	0	0	0	0	+4	60%
Masking										
Voting	-1	-4	0	0	+4	+4	0	+1	-2	100%
Relationships	P	N	P	P	P	P	N	N	P	
Impact Median Value	0	0	+1	+1.5	0	+0.5	-1	-0.5	+1	

Table 10: Maintainability Candidate Architectural Impact Evaluation

Dependability Quality Attribute (Maintinability) | Impact Evaluation on other Dependability & Embedded Systems Quality Attributes										
Maintainability Candidate Architectural Tactics	Functionality	Efficiency	Usability	Portability	Availability	Reliability	Integrity	Confidentiality	Safety	Level of Agreement
Manage Input/Output										
Record/Playback	0	0	0	0	+1	+1	0	-2	+2	80%
Separate Interfaces from Implementation	0	0	0	+4	+2	+2	0	0	+2	100%
Specialized Access Routines /Interfaces	0	+4	0	-1	0	0	-1	0	0	60%
Localize Changes										
Semantic Coherence	0	-2	0	+4	0	0	0	0	0	80%
Anticipate Changes	0	-2	0	+4	0	0	0	0	0	80%
Generalize Module	0	-1	0	+2	0	0	0	0	0	100%
Limit Possible Options	-2	+1	-2	0	0	0	0	0	+1	60%
Abstract Common Services	0	-2	0	+4	0	0	0	0	0	80%
Prevention of Ripple Effect										
Hide Information	0	-2	0	+4	0	0	0	0	0	100%
Maintain Existing Information	0	-1	0	+2	0	0	0	0	0	80%
Restrict Communication Paths	0	+2	0	+1	0	0	+3	0	0	60%
Use an Intermediary	0	-4	+2	+2	+4	+4	0	-1	+3	80%
Internal Monitoring										
Built-in Monitors	0	-1	+2	-3	+4	+4	+4	-2	+4	100%
Defer Binding Time										
Run-Time Registration	0	-4	0	+2	0	0	0	0	0	80%
Configuration Files	+2	0	+2	0	0	0	-2	-1	0	100%
Polymorphism	0	-2	0	0	0	0	0	0	0	80%
Component Replacement	+4	-2	+2	+2	0	0	0	-2	0	100%
Adherence to Define Protocols	+2	+2	0	+3	+1	0	0	-1	0	60%
Relationships	P	N	P	P	P	P	P	N	P	
Impact Median Value	+1	0	0	+0.5	+2	+2	+1	-1	+2	

Table 11: Quality Attributes Relationships found in [1] and [38]

Quality Attribute	Functionality	Efficiency	Reliability	Usability	Maintainability	Portability
Efficiency	Negative		Independent	Negative	Negative	Negative
Reliability	Positive	Independent		Positive	Negative	Independent
Usability	Positive	Negative	Positive		Independent	Independent
Maintainability	Positive	Negative	Positive	Independent		Positive
Portability	Independent	Negative	Independent	Independent	Positive	

5 Analysis

This research is designed based on two main objectives: The first one is to understand the nature of the inter-relationship between dependability quality attributes and other quality attributes of embedded systems based on the architectural solution provided by the software architecture community. The second objective is to increase embedded systems' dependability by evaluating the impact of using tactics in order to achieve the dependability of embedded systems.

Research results indicate that the relationships between dependability quality attributes are interchanging. For example, when achieving the dependability quality attribute maintainability by using proposed architectural tactics, this has a slight support on achieving the dependability quality attributes integrity and confidentiality and therefore create a positive relationship. When focusing on achieving the dependability quality attributes integrity and confidentiality, the use of the proposed architectural tactics has a minor impact which hinders the achievement of dependability quality attribute maintainability and therefore creates a negative relationship.

There are many tactics that can be used to satisfy a certain concern of a dependability quality attribute. Such tactics have different levels of impact on the achievement of dependability and other embedded systems quality attributes. Software architects have the option to select and calibrate appropriate architectural solutions or tactics to satisfy a certain concern for a dependability quality attribute and therefore balance the relationship between dependability and other embedded system quality attributes to meet certain system specifications. For example, to achieve the dependability quality attributes availability and reliability and satisfy one of its concerns "Recovery Preparation and Repair", software architects are recommended to select the tactic "Passive Redundancy" . The impact of using such a tactic will deter less the achievement of dependability and other embedded systems quality attributes. On the other hand, the selection of the architectural tactic "Active Redundancy" constrains the achievement of other dependability and embedded systems quality attributes.

Moreover, the median value of impact level of proposed architectural tactics represents the strength of the relationship between dependability and other quality attributes of embedded systems. For example, when utilizing the proposed tactics to achieve the dependability quality attribute "Safety" the impact median value shows to what level or extent it supports or hinders the achievement of dependability and other embedded systems quality attributes according to the predefined scale for impact level.

Balancing the relationships between dependability and embedded system quality attributes is done in two steps. First, by evaluating the impact level of the utilized architectural tactics to achieve a dependability quality attribute on other dependability and embedded systems quality attributes achievement. Secondly, by selecting the appropriate architectural tactic that helps in sustaining a high level of dependability and quality of embedded systems.

It is important to understand that the adoption of a certain architectural tactic used to achieve a dependability quality attribute of a system can affect the relationship between dependability and other embedded system quality attributes. For example, the candidate architectural tactics of the dependability quality attributes "integrity and confidentiality" deter the chances of achieving the dependability quality

attribute "maintainability" quality criteria "testability" by adding complexity to the system.

Moreover, the dependability quality attribute "safety" supports the achievement of the quality attribute "portability" of embedded systems, because safety candidate architectural tactics aim at maintaining the quality sub-characteristics of the quality attribute "portability" for the system by making the system adhere to application related standards. Additionally, the safety candidate architectural tactics can also be adjusted according to the systems' quality attribute "portability" conventions or regulations and similar prescriptions.

By comparing the results of the aforementioned recent studies with this study's results, it can be seen that the relationships of quality attributes can be assessed differently. In this research, the relationship between quality attributes is based on the used solutions that are adopted to achieve a certain quality attribute. In the other research studies, the results are more generalized and are based on experiences from different views.

6 Discussion

Dependability is usually left until the late stages of software development. This research focuses on manifesting dependability requirements and concerns at earlier stages such as at the software architecture stage. In this stage we have studied the methodologies that can be used to achieve dependability quality attributes. One of the quality engineering methodologies and approaches to achieve dependability quality attributes to an embedded systems is the architectural strategy "tactics".

The understanding of quality attributes relationships one of the factors that steer software development toward success. Therefore, we have developed a framework to understand the relationships of quality attributes based on the impact evaluation of used tactics and relevant software quality requirements. We have taken dependability quality attributes separately and evaluated their candidate solutions' impact on other quality attributes.

The quality attributes' reasoning framework is helpful to evaluate the quality attributes' impact in case a solution exists. This framework can only assess the impact when there are enough specifications of the developed system and its quality requirements. It also gives the opportunity to assess the architectural tactics concerning the system's desired quality attributes before using them. Moreover, the framework may produce a neutral relationship between the quality attribute in focus and the compared quality attributes when the number of the candidate architectural tactics of the quality attribute in focus which have a positive impact are equal to those which have a negative impact on the compared quality attributes.

Furthermore, the dependability of embedded systems can be increased by selecting the appropriate architectural design decisions which have supportive influences on other dependability and embedded systems quality attributes. Additionally, this framework helps to define what the quality criteria are which can be affected by the other quality attributes of a system. For example, achieving the dependability quality attributes integrity and confidentiality by using the tactic "Active Redundancy" hinders the achievement of testability and therefore hinders the achievement of maintainability and creates a negative relationship.

The degree which describes the level of impact is dependent on the extent to what a given dependability quality attribute candidate architectural tactic affects the achievement of other quality attributes' sub-characteristics (criteria)- as most of the participants of the research's second survey wrote in their motivations explaining the base of their evaluation. For example, the candidate architectural tactics that try to satisfy the concern of "Availability and Reliability" for failure detection have a great influence on the embedded system's quality attribute "Functionality" supporting the achievement of all the sub-characteristics of the "Functionality" quality attribute and thus creates a positive relationship with the embedded system's quality attribute "Functionality" when satisfying the "Availability and Reliability's" fault detection concern. On the other hand, most of the "Availability and Reliability" candidate architectural tactics used to satisfy recovery preparation and repair create various levels of negative

relationships and one positive relationship with embedded systems quality attribute "Efficiency". Thus, tactics impact evaluation can be undertaken at two levels:

- As quality attributes consist of several sub-characteristics [24]. The first level describes the extent of hindering or supporting the compared quality attributes' sub-characteristics/ criteria when using a given tactic. For example, using the tactic "Authenticate Users" to satisfy the concern "Resisting Attack" of the dependability quality attributes "Integrity and Confidentiality" assists the achievement of the embedded system's quality attribute "Usability" by supporting its sub- characteristics/ criteria "Operability" and "Understandability".
  The use of the same tactic will on the other hand hinder the achievement of the embedded system's quality attribute "Portability" by preventing the achievement of its sub-characteristics "Ease of Integration" and "Compatibility".

- The Second level of impact evaluation is the level by which this tactic is hindering/supporting the achievement of the compared quality attributes. For example, the achievement of the dependability quality attributes' "Availability and Reliability" concern "Recovery Preparation and Repair" via the proposed tactics has different levels of negative and positive impact over the embedded system's quality attribute "Efficiency".
  The variety between the levels in this case - as the participants provided multiple reasons to motivate their evaluation- is dependent on the amount of "cost in time and resources" of the embedded systems. As the tactic "Passive redundancy" does not consume more money than other proposed tactics for the same concern - as it's assigning one component to do the work rather than constructing another copy or redundant system components to perform multiple tasks -this tactic has a minor positive impact on achieving efficiency for embedded systems as it does not support the achievement of all the sub-characteristics of the embedded system's quality attribute "Efficiency".

6.1 Validity

The demonstration of the results' validity is based in this research on the possible validity threats discussed by [40]. In this research, the possible internal validity threats are carefully treated. A stratified method as the sampling method has been has been used as it reduces the amount of errors in the process of choosing study subjects/ participants [37]. Moreover, the author classified subjects/participants into different single groups according to their profiles considering the possible validity threats to single groups denoted by [C. Wohlin et al.] such as "Statistical Regression and Selection". The validity threats for these single group are carefully estimated, observed, assessed, and treated.

In this research we have used a survey as an approach to collect qualitative data to enable the researchers to produce quantitative data and thus generate indicators identifying the nature and strength of the relationships between a specific dependability quality attribute with other dependability quality attributes and embedded systems' quality attributes based on the impact evaluation of the proposed dependability quality attribute tactics. Construct and external validity threats of this study consist in short of two parts:

The validity of translating embedded systems' quality attributes to a higher level and the validity of evaluating software architectural tactics. Starting with the last, we have adopted a careful approach when combining different evaluations of the dependability quality attributes candidate architectural tactics. We have carefully examined the motivations and bases that the participants considered when provided their evaluations for the architectural tactics. In conclusion, if any doubt can be casted on the results, we argue that this will be in the form of the different base for evaluating software architecture candidate tactics. Appendix C.2 describes in details the research survey "Dependability Quality Attributes Candidate Architectural Tactics Impact Evaluation ".

Translating embedded systems' quality attributes to a higher level using a quality model was achieved by selecting the ISO 9126 quality model. The ISO 9126 quality model is globally accepted and provides

the necessary details to enhance the translation process. More information regarding the ISO 9216 quality model can be found in [24]. Moreover, in order to assess the adequacy of the translation process we conducted a survey that was distributed among software engineers to collaborate in giving the right translation of low-level embedded systems' quality attributes into ISO 9126 counterparts. The translation details of the embedded systems' quality attributes can be found in Appendix C.1.

7 Conclusions

When developing a software system, it is important to understand the impact of the used strategy or tactic on other desired quality attributes at earlier stages. Otherwise, more effort may be spent on trying to satisfy the system requirements.

Conversely, the system development maybe facilitated if it is known that by satisfying a quality attribute using a certain architectural tactic, other quality attributes are likely to follow because of the positive impact the used architectural tactic has.

Hence, there is a need for an increased understanding of the relationships between quality attributes when using the different tactics and strategies. In this research, we have studied the relationships between dependability quality attributes with the help of architectural tactics. The research aims at exploring the solutions that already exist in order to increase the dependability for embedded systems. We have mapped candidate tactics that can be used to achieve different dependability quality attributes. We have also performed a rigorous assessment to identify the architectural tactics' impact on other dependability quality attributes and on embedded system quality attributes.

This research provides indicators to define the relationship between dependability and other quality attributes of embedded systems by quantifying the impact of available solutions. The two main contributions of this research are focused on increasing the likelihood of having more dependable embedded systems by providing thorough analysis for the dependability quality attributes' relationship based on the available solution that could be illustrated from literature and experience. This research results provide yet another factor for software architects to take into account when deciding on a dependability quality attributes candidate architectural tactics prioritisation; what, and in what order.

We have developed a framework to understand the relationships between quality attributes based on the used solutions and systems' quality requirements. The quality attribute relationships reasoning framework helps in understanding the importance of making the right design decision embracing system quality requirements and other constraints. The evaluated impact of these solutions will help in making the proper selection of tactics to achieve dependable embedded systems and therefore satisfy one of the embedded systems most important requirements. Moreover, we conclude that, when utilizing a tactic for developing a system, software architects must take into account to evaluate their architectural tactics or strategies according to different aspects such as impact on achieving dependability and other embedded systems quality attributes, cost, implementation applicability to system context and other factors as well.

7.1 Future Work and Recommendations

Further studies are necessary to understand the reasons of the influence that a candidate dependability quality attributes architectural tactic have on the achievement of dependability and other embedded systems quality attributes. Further lines of research could be in the prioritisation of dependability quality attributes candidate architectural tactics with respect to other dependability or embedded systems quality attributes achievement.

References

[1] M. Svahnberg and K. Henningsson, "Consolidating different views on quality attributes relationships," in *Proceedings of the 8th Software Engineering Research and Practice (SERP)*, 2008.

[2] J. A. McCall, "Quality factors," *Encyclopedia of Software Engineering (Marciniak, J., ed.)*, pp. 958–969, 1994.

[3] K. Misra, *Handbook of Performability Engineering*, 1st ed. Germany: Springer, 2008, ch. Dependability Considerations in the Design of a System, pp. 71–80.

[4] H. V. Vliet, *Software Engineering Principles and Practice*, 2nd ed. New York: John Wiley, 2000, ch. On Managing Software Quality, pp. 101–126.

[5] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed. New York: Addison Wesley, 2003, ch. Understanding Quality Attributes and / Achieving Quality, pp. 71–127.

[6] C. Ebert and R. Dumke, *Software Measurement: Establish-Extract-Evaluate-Execute*. New York - London -Springer: Springer, 2007, ch. Assuring the Quality of Software Systems, pp. 230–300.

[7] C. Andersson, "Managing software quality through empirical analysis of fault detection," Ph.D. dissertation, Lund University,Dept. of Communication Systems ,Dissertation (No. 1101-3931), Skåne-Lund, Sweden, 2006.

[8] I. of Electrical &Electronic Engineers, "IEEE standard glossary of software engineering terminology," IEEE, Tech. Rep. IEEE Std. 610.12-1990, 2002.

[9] E. A. Strunk and J. C. Knight, "Dependability through assured reconfiguration in embedded system software," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 3, pp. 172–187, 2006.

[10] M. Larsson, "Predicting quality attributes in component-based software systems," Ph.D. dissertation, Mälardalen University, Department of Computer Science and Electronics, 2004.

[11] I. Crnkovic, "Component-based approach for embedded systems," in *In Proceedings of 9 International Workshop on Component-Oriented Programming*, 2004.

[12] A. Vulgarakis and C. Seceleanu, "Embedded systems resources: Views on modeling and analysis," *Computer Software and Applications Conference, Annual International*, vol. 0, pp. 1321–1328, 2008.

[13] I. Lee, J. Leug, and S. Son, *Handbook of Real-Time and Embedded Systems*. New York: Taylor and Francis Group, 2008, ch. Dynamic QoS Management in Distributed Real-Time Embedded Systems, pp. 36–1/12.

[14] K. Misra, *Handbook of Performability Engineering*. Germany: Springer, 2008, ch. Dependability Consideration in the Design of a System, pp. 71–80.

[15] M. Barbacci, M. H. Klein, T. A. Longstaff, and C. B. Weinstock, "Quality attributes," Software Engineering Institute Carnegie Mellon University, Pennsylvania, USA, Tech. Rep. ESC-TR-95-021, 1995.

[16] J. Heit, "Impact of methods and mechanisms for improving software dependability on non-functional requirements," Master's thesis, Stuttgart University, Dept. Informatics , No.(2548), Stuttgart, Germany, 2007.

[17] C. Gacek and R. de lemos, *Structure for Dependability Computer Based Systems from Interdisciplinary Perspective*. London: Springer, 2006, ch. Architectural Description of Dependable Software Systems, pp. 127–138.

[18] T. Sherman, *Advances in Computer and Information Sciences and Engineering*. Germany: Springer, 2008, ch. Quality Attributes for Embedded Systems, pp. 535–539.

[19] A. Avizienis, J. Laprie, and B. Randell, "Fundamental concepts of dependability," in *Proceedings of the 3rd International Survivability workshop*, 2001.

[20] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE*

Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11–33, 2004.

[21] J. C. Laprie, "Dependable computing and fault tolerance : Concepts and terminology," in *Fault-Tolerant Computing, 1995, ' Highlights from Twenty-Five Years', Twenty-Fifth International Symposium on*, 1995, pp. 2–8.

[22] L. Zhu, M. A. Babar, and R. Jeffery, "Mining patterns to support software architecture evaluation," *Software Architecture, Working IEEE/IFIP Conference on*, vol. 0, p. 25, 2004.

[23] M. S. Deutsch and R. R. Willis, *Software Quality Engineering a Total Technical and Management Approach*. New Jersey: Prentice Hall, 1988.

[24] P. Berander, L. Damm, J. Eriksson, T. Gorschek, K. Henningsson, P. Jönsson, S. Kågström, D. Milicic, F. Mårtensson, K. Rönkkö, P. Tomaszewski, L. Lundberg, M. Mattsson, and C. Wohlin, "Software quality and trade-offs," Blekinge Institute of Technology, Ronneby, Blekinge- Sweden, Tech. Rep. Tech. 2005, June 2005.

[25] L. Bass, M. Klein, and G. Moreno, "Applicability of general scenarios to the architecture tradeoff analysis method," Carnegie Mellon University, Pittsburgh:PA, USA, Tech. Rep. TR-014, Oct 2001.

[26] M. Svahnberg and C. Wohlin, "A comparative study of quantitative and qualitative views of software architectures," in *Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering*, 2003, pp. 1–8.

[27] N. Lassing, D. Rijsenbrij, and H. van Vliet, "On software architecture analysis of flexibility, complexity of changes size isn't enough," in *Proceedings of Second Nordic Software Architecture Workshop (NOSA '99)*, 1999, pp. 1103–1581.

[28] R. Kazman, S. J. Carrière, Woods, and G. Steven, "Toward a discipline of scenario-based architectural engineering," *Ann. Softw. Eng.*, vol. 9, no. 1-4, pp. 5–33, 2000.

[29] W. Wu and T. Kelly, "Safety tactics for software architecture design," *Computer Software and Applications Conference, Annual International*, vol. 1, pp. 368–375, 2004.

[30] F. Bachmann, L. Bass, and R. Nord, "Modifiability tactics," Software Engineering Institute Carnegie Mellon University, Pittsburgh:PA, USA, Tech. Rep. TR-002, Oct 2007.

[31] J. Karlsson, C. Wohlin, and B. Regnell, "An evaluation of methods for prioritizing software requirements," *Information and Software Technology*, vol. 39 (14-15), pp. 938–947, nov 1999.

[32] M. Svahnberg and C. Wohlin, "An investigation of a method for identifying a software architecture candidate with respect to quality attributes," *Empirical Softw. Engg.*, vol. 10, no. 2, pp. 149–181, 2005.

[33] M. Svahnberg, C. Wohlin, L. Lundberrg, and M. Mattsson, "A quality driven decision support method for identifying software architecture candidates," *International Journal of Software Engineering and Knowledge Engineering*, vol. 13(5), pp. 547–573, 2003.

[34] M. Svahnberg, "Supporting software evolution: Architecture selection and variability," Ph.D. dissertation, Blekinge Institute of Technology,Dept Software Engineering and Computer Science ,Dissertation(No.2003:03), Ronneby - Blekinge, Sweden, 2003.

[35] J. Bosch and P. Molin, "Software architecture design: Evaluation and transformation," *Engineering of Computer-Based Systems, IEEE International Conference on the*, vol. 0, p. 4, 1999.

[36] Swedish-Space-Cooperation. (2009) Smart 1-2005 technical information,(available online on:http://www.ssc.se/?id=6217), retrieved on: April. [Online]. Available: http://www.ssc.se/?id=6217

[37] W. G. Cochran, *Sampling techniques*, 3rd ed. New York: Wiley, 1977.

[38] K. Henningson and C. Wohlin, "Understanding the relations between software quality attributes- a survey approach," in *in Proceedings of the 12th International Conference for Software Quality*, 2002.

[39] H. Zulzalil, A. A. A. Ghani, M. H. Selamat, and R. Mahmod, "A case study to identify quality attributes relationships for web-based applications," in *International Journal of computer Science and Network Security (IJCSNS)*, vol. 8(11), 2008, pp. 215–220.

[40] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering. An Introduction*, 1st ed. Massachuttes:USA: Kluwer Academic Publisher, 2000, ch. Planning, pp. 47–73.

[41] R. Czaja and J. Blair, *Designing Surveyes: a guide to decision and procedures*, 2nd ed. Thousand Oaks-California: USA: Sage Publications:Pine Fore Press, 2005, ch. Stages of a Survey, pp. 11–32.

A Dependability and Embedded Systems Quality Attributes General and Concrete Scenarios

A.1 Embedded Systems Quality Attributes General Scenarios

General Scenarios are distilled from the common requirements placed on embedded systems software and dependability quality attributes concerns. In order to generate general scenarios from quality requirements, we will assume the embedded system is a satellite system. We have selected a satellite system as its characteristics fulfill the general requirements and its performed services are combining different embedded system services or missions (e.g. communication, long life operation time, etc.). There are several matrixes used to present quality attributes. However, this research will consider the matrix suggested by [5]. The following are general scenarios for the translated embedded systems quality attributes.

- Functionality

  According to stakeholder (Source of Stimulus) requirements the satellite system both hardware and software components (Artifact) must be supported by multiple functions and features that are needed to accomplish its mission requirements (Stimulus), be accurate in providing the expected results, and to be adhere (Response) to successfully fulfill the predefined design specifications (Response Measure) and orbiting the moon (Environment). Table 12 shows functionality general scenario.

- Efficiency

  There are a large number of task the satellite system is in need to handle (Stimulus). The satellite system development has constraints on its overall weight and response time processing tasks (Source of Stimulus). Due to different environments where the system will go through (Environment); the satellite system (Artifact) is required to perform these tasks (Response) meeting the agreed minimum usage rate of its resources, and a minimum load rate on the resource being used to execute a task (Response Measure). Table 13 shows efficiency general scenario.

Elements	Description
Source of Stimulus	Stakeholders
Stimulus	Multiple of functions and features needed to accomplish the results
Simulated artifacts	Satellite Software System
Environment	Normal Condition
Response	Functions and Features are accurate in producing results, meets the mission requirements, adhere with system design specifications
Response Measure	Fulfill the mission goal, accurate , and meets technical and business regulations.

Table 12: Functionality General Scenario

Elements	Description
Source of Stimulus	System development Specification
Stimulus	Large number of tasks
Simulated artifacts	Satellite Software System
Environment	Normal Condition
Response	Execute different tasks within deadline
Response Measure	Small amount of resources used, and minimum load on used resources.

Table 13: Efficiency General Scenario

- Usability

  The satellite system is going to be controlled and used from the control-base (Source of Stimulus). The system is handled by multiple of control-base staff where a large amount of tasks performed remotely from the satellite system itself (Stimulus). During its operation orbiting the moon (Environment) the system (Artifact) shall provide the opportunity to distinct views with consistent operation which give the End-

Elements	Description
Source of Stimulus	Satellite Control Base
Stimulus	Use the system efficiently and minimize the impact of errors.
Simulated artifacts	System Hardware and Software Components.
Environment	Normal Condition
Response	System provide distinct views with consistent operation and therefore ease to operate system functionalities
Response Measure	Task execution time and the amount of errors.

Table 14: Usability General Scenario

Elements	Description
Source of Stimulus	Organization business and technical regulations
Stimulus	Reuse of System components.
Simulated artifacts	Software System Components.
Environment	Deactivated
Response	System provide distinct views with consistent operation and therefore ease to operate system functionalities
Response Measure	Time to install them to the new environment and amount of modification needed to be implemented to the component

Table 15: Portability General Scenario

Users ease of Operating the system and using its functionalities to execute a task (Response) while end-users are expected to be able to interact with the system after completing training course with a minimum amount of errors (Response Measure).
Table 14 shows general scenario for embedded systems usability quality attribute.

- Portability

According to development organization regulations (Source of Stimulus) that are one of its objectives is to reuse components (Stimulus) when the satellite mission is over. Therefore, satellite hardware and software components (Artifact) are required to be easily integrated into the satellite platform and have the property by which it allows the organization to use these components when needed. (Response) This requirement shall be done within a minimum amount of effort and modifications (Response Measure).
Table 15 shows the general scenario for embedded systems usability quality attribute.

A.2 Embedded Systems Quality Attributes Concrete Scenarios

Although general scenarios provide a characterization of the quality attributes, quality attributes reasoning framework is still in need to have more detailed levels. In the previous section we have generated quality attributes general scenarios for a satellite system. In this section we will generate embedded systems quality attributes and dependability quality attribute concrete scenarios by consolidating general scenarios with more specification and a higher scale level of details about this satellite system.

We will employ resources from real-life using some of the technical information provided by the Swedish Space Corporation "SSC" about the satellite "SMART 1- 2005" [36]. The following are the concrete scenarios for quality attributes:

-Functionality

Scenario I (Accuracy)

There are a large number of requests arriving to SMART 1-2005 (stimulus) from different resources from the satellite control-base (source of stimulus) during its operation orbiting the moon (environment). For the satellite system both components hardware and software (artifact) are required to process these tasks and return 100% accurate results (response). The accuracy of the returned results is compared to design specification (response measure).

Table 16 shows the concrete scenario for functionality quality attribute sub-characteristics (Accuracy).

Scenario II (Security)
SMART 1-2005 project risk management recommends that the satellite system will be subjected

Elements	Description
Source of Stimulus	Satellite Control Base
Stimulus	large number of requests arriving to SMART 1-2005.
Simulated artifacts	System Hardware and Software Components.
Environment	Normal Condition
Response	process requests and return results within specific period of time
Response Measure	The accuracy of the returned results is compared to design specification.

Table 16: Accuracy Criteria Concrete Scenario

Elements	Description
Source of Stimulus	Different Resources (Environmental, Intruders, etc...).
Stimulus	various types of attacks.
Simulated artifacts	Satellite Software System.
Environment	Normal Condition
Response	preventing different attacks to the system%
Response Measure	high capability handling all different expected attacks.

Table 17: Security Quality Criteria Concrete Scenario

to various types of attacks (stimulus) from different resources (source of stimulus), for example interruption of the communication signal. System technical specifications require the satellite software system (artifact) to have a high capability (response measure) in preventing different attacks to the system (response) and therefore secure system components functionalities throughout its operation orbiting the moon (environment).
Table 17 shows the concrete scenario for functionality quality attribute sub-characteristics Security.

Scenario III (Expansion Capability)

SMART 1-2005 satellite mission objectives may require additional features/ functionalities to be enclosed to its system (stimulus) during its service time and when the project is being suspended (environment). System developers (source of stimulus) are responsible to build additional features or functionalities for the satellite system (artifact). Both satellite system components hardware and software must provide consistent and simple logical and physical structures (response). Adding features/ functionality to a satellite system must not take the developer more than a week (response measure).
Table 18 shows the concrete scenario for functionality quality attribute sub-characteristics Expansion Capability.

Elements	Description
Source of Stimulus	Source of Stimulus
Stimulus	additional features/ functionalities to satellite system.
Simulated artifacts	Satellite System hardware and software components.
Environment	Normal Condition
Response	Provide consistent and simple logical and physical structure
Response Measure	Adding features or functionalities to satellite system shall not exceed one week.

Table 18: Expansion Capability Concrete Scenario

- Efficiency

- Resources Behavior:

Scenario I (Code Size)

System design specification (source of stimulus) constraints SMART 1-2005 software to be efficient. The system software is required to have a fixed size of code (stimulus). Therefore, system developers must not exceed (response) the amount of 1 million lines of code (LOC) (response measure). This constraint is applied to the system software (artifact) development or to the upgrade during its operation time (environment).
Table 19 shows the concrete scenario for efficiency quality attribute sub-characteristics resource behavior (code size).

Scenario II (Memory Usage)

Satellite system technical rules (Source of stimulus) constrains the satellite system to use memory which have the following properties: 2MB EERPROM permanent storage, 3MB static RAM EDAC, 4GB

Elements	Description
Source of Stimulus	Design requirements
Stimulus	Fixed size of code.
Simulated artifacts	Satellite system software.
Environment	Normal Condition
Response	Must be less than 1 million lines of code
Response Measure	Size of line of code (SLOC).

Table 19: Code Size Concrete Scenario

Elements	Description
Source of Stimulus	Technical and Design requirements
Stimulus	Fixed type and size memory.
Simulated artifacts	Satellite system.
Environment	Normal Condition
Response	Satellite sytem shall not violate technical and design specification on used memory
Response Measure	Each task satellite system performs shall not exceed 0.1% of memory usage.

Table 20: Memory Usage Concrete Scenario

mass memory Samsung DRAM (Artifact). There are a large number of tasks the system has to process (Stimulus) during its operation orbiting the moon (Environment). Due to the design specifications, the system is expected to use its memory efficiently. Each task the system has to process shall not exceed (Response) the amount of 0.1% of memory usage (Response measure).
Table 20 shows the concrete scenario for efficiency quality attribute sub-characteristics resource behavior (Memory Usage).

Scenario III (Complexity, CPU, Peripherals)

SMART 1- 2005 system architects (source of stimulus) decided to build the system on a 3-axis stabilized platform (Environment). The system (Artifact) shall have a single-chip processor, sensors, actuators, antennas and 41 power switches providing an over-current/under voltage protection (Stimulus). However, these components shall not violate (Response) the simplicity constraints, response time, and consistency in processing multiple tasks (Response Measure).
Table 21 shows the concrete scenario for efficiency

Elements	Description
Source of Stimulus	Satellite system architectures
Stimulus	the system shall have a single-chip processor, sensors, actuators, antennas and 41 power switches providing an over-current/under voltage protection.
Simulated artifacts	Satellite system.
Environment	Developing Stage
Response	System shall not violate the simplicity requirements, response time, and consistency in processing multiple tasks
Response Measure	Development and operation requirements.

Table 21: Complexity, CPU, and Peripherals Concrete Scenario

quality attribute sub-characteristics resource behavior (Complexity, CPU, Peripherals).

Scenario IV (Resources Used)

The satellite launch requirements (Source of Stimulus)constrains satellite system to carry no more fuel than what is enough for an effective and successful launch of <350 kg mass to the space (Stimulus). The SMART 1-2005 system consists of different hardware components with different properties. System architects are requested to consider the hardware components weight when building the satellite (Artifact). The overall weight of SMART 1-2005 must not exceed (Response) a maximum of 349 kg of total mass during its development (Environment).
Table 22 shows the concrete scenario for efficiency quality attribute sub-characteristics resource behavior (Resources Used).

Scenario V (Physical Size)
The satellite body shape is obligated to obey launching and operation rules (Source of Stimulus). The SMART 1-2005 system architects request to implement the SMART 1-2005 (Artifact) within a three dimension (Stimulus) of $3m \times 3m \times 4m$(Response measure). The overall shape of the system is expected to be efficiently complying with theses constraints (Response) to have less air fraction while

Elements	Description
Source of Stimulus	Satellite launch requirements
Stimulus	effective and successful launch of <350 kg mass to the space.
Simulated artifacts	Satellite system hardware and software components.
Environment	Development Stage
Response	Resources used in satellite development shall comply with development and launch requirements efficiently
Response Measure	satellite weight > 340kg.

Table 22: Resources Used Concrete Scenario

Elements	Description
Source of Stimulus	Satellite launch and operation requirements
Stimulus	Developing satellite system in a three dimensional shape.
Simulated artifacts	Satellite system hardware components.
Environment	Development Stage
Response	Resources used in satellite development shall comply with launch and operation requirements
Response Measure	Satellite shape success in having three dimensional shape with distance of $3m \times 3m \times 4m$.

Table 23: Physical Size Concrete Scenario

launching from or returning to earth (Environment). Table 23 shows the concrete scenario for efficiency quality attribute sub-characteristics resource behavior (Physical Size).

Scenario VI (Power Consumptions)
The specification of SMART 1-2005 system operations (Source of Stimulus) requires the system to store/ use solar 50 KW energy (Stimulus) in a duration of full orbit around the moon (Environment) and therefore the system software and hardware components (Artifact) are expected to have a low power consumption (Response) that is estimated to be less than what the system can store during half orbit (Response Measure) in its normal condition.
Table 24 shows the concrete scenario for efficiency quality attribute sub-characteristics resource behavior (Power Consumption).

- *Time Behavior:*

Elements	Description
Source of Stimulus	Satellite operation requirements
Stimulus	Store/use 50 kw solar energy.
Simulated artifacts	Satellite system hardware and software components.
Environment	Normal Condition
Response	System components shall have low power consumption
Response Measure	Power consumption shall be less that what a satellite can store in the duration of half orbiting the moon.

Table 24: Power Consumption Concrete Scenario

Elements	Description
Source of Stimulus	Design Requirements
Stimulus	large amount of tasks.
Simulated artifacts	Satellite Hardware (CPU).
Environment	Normal Condition
Response	Execute tasks on the CPU within a certain amount of load time
Response Measure	Maximum load time on the system CPU is 1 % per each task or process.

Table 25: Physical Size Concrete Scenario

Scenario I (CPU Time Used)

According to the design specifications (Source of stimulus) SMART 1-2005 is required to use an ERC 32, 20MHz single-Chip processor. There is a large amount of tasks the system CPU (Artifact) is required to process (Stimulus). Each of these tasks is executed on the CPU within a certain amount of load time (Response). Maximum load time on the system CPU is 1 % (Response measure) per each task or process while operating in its orbit (Environment).
Table 25 shows the concrete scenario for efficiency quality attribute sub-characteristics time behavior (CPU Time Used).

Scenario II (Performance)

A large number of requests (Stimulus) on the SMART 1-2005 system from the control-base (Source of Stimulus) arrive at the system in normal condition while orbiting the moon (Environment). SMART 1-2005 (Artifact) has to execute these requests and transfer (Response) results with a maximum time of

Elements	Description
Source of Stimulus	Satellite control base
Stimulus	large amount of tasks.
Simulated artifacts	Satellite Software.
Environment	Normal Condition
Response	Execute and transfer results
Response Measure	maximum time of 0.1 second per request.

Table 26: Performance Concrete Scenario

Elements	Description
Source of Stimulus	Satellite control base
Stimulus	Using satellite features and functions.
Simulated artifacts	Satellite System hardware and software.
Environment	Normal Condition
Response	Ease of understanding and adapting to learn how system feature and functions work
Response Measure	No longer than a week.

Table 27: Understandability Concrete Scenario

0.1 second (Response Measure).
Table 26 shows the concrete scenario for efficiency quality attribute sub-characteristics time behavior (Performance).

- Usability

Scenario I (Understandability)

SMART 1-2005 is handled by the satellite control-base. The control-base has divided tasks up into its different departments (Source of Stimulus). Each of the department's staff who wants to use (Stimulus) the system's features/ functionalities (Artifact) under normal condition (Environment) should be able to adapt to the system features after (Response) finishing a training course. The user shall feel comfortable using the system and will be able to learn how to use its feature within one working week(Response Measure).
Table 27 shows the concrete scenario for usability quality attribute sub-characteristics understandability.

Scenario II (Learn-ability)

Elements	Description
Source of Stimulus	Satellite control base new staff
Stimulus	Learn system functions and features.
Simulated artifacts	Satellite System hardware and software.
Environment	Normal Condition
Response	Ease of learning and controlling to system functions and features
Response Measure	Minimum amount of errors.

Table 28: Learn-ability Concrete Scenario

Control-base staff (Source of Stimulus) handling SMART 1-2005 system (Artifact) is required to be able to learn how to use the system throughout its operation time (Stimulus). The system features and functionalities shall provide the opportunity for the users to adapt to the system (Response) and therefore have a minimum amount of use errors (Response Measure) in normal operation conditions (Environment).
Table 28 shows the concrete scenario for usability quality attribute sub-characteristics understandability.

Scenario III (Operability)

During the time that the satellite is orbiting the moon, the SMART 1-2005 control -base staff (source of stimulus) is expected be able to efficiently (response) use (stimulus) the satellite system operations (artifact). The impact of operation control errors should be minimized by the efficient use of SMART 1-2005 in normal conditions (environment)
Table 29 shows the concrete scenario for usability quality attribute sub-characteristics Operability.

- Portability

Scenario I (Ease of Integration)

The SMART 1-2005 system (Artifact) has a large number of hardware and software components (Stimulus) that are selected to build the satellite according to its specifications (Source of Stimulus). The satellite components are required to provide the

Elements	Description
Source of Stimulus	Satellite control base staff
Stimulus	efficiently to operate satellite system functions and features.
Simulated artifacts	Satellite System hardware and software.
Environment	Normal Condition
Response	Efficiently operating system features and functions
Response Measure	impact of errors when operating satellite system features shall be minimized.

Table 29: Operability Concrete Scenario

Elements	Description
Source of Stimulus	Satellite Development and Operation Specifications
Stimulus	Satellite components integration to system platform.
Simulated artifacts	Satellite System Components.
Environment	Normal Condition
Response	Components able and allow to integrate with satellite platform
Response Measure	successfully integration.

Table 30: Ease of Integration Concrete Scenario

opportunity (Response) to be successfully integrated to the system platform according to its predefined specifications (Response Measure) during its development life cycle (Environment).
Table 30 shows the concrete scenario for portability quality attribute sub-characteristics Ease of Integration.

Scenario II (Compatibility)

The Satellite SMART 2005-1 might be required to be transferred to another satellite platform (Stimulus). The satellite will receive the new modification information from the control base (Source of Stimulus). Therefore the system hardware and software components (Artifact) are expected to follow new coordination data and accept the proposed modifications to the new platform(Response). Moreover, the satellite must not require a large amount of effort to be modified in order to be compatible with the new platform. In the contrary the system is expected to be able to adapt to the new modifications fast to its

Elements	Description
Source of Stimulus	Satellite control base
Stimulus	transfer to another platform.
Simulated artifacts	Satellite System Components.
Environment	Normal Condition
Response	Components capable of changing platforms.
Response Measure	low modifications and successful adaption to the new platform.

Table 31: Compatibility Concrete Scenario

Elements	Description
Source of Stimulus	Design Requirements
Stimulus	Handling failures.
Simulated artifacts	Satellite System.
Environment	Normal Condition
Response	System shall recuperate from disruption.
Response Measure	low modifications and successful adaption to the new platform.

Table 32: Reliability General Scenario

features (Response Measure) to allow the satellite to adapt to a new platform (Environment).
Table 31 shows the concrete scenario for portability quality attribute sub-characteristics compatibility.

A.3 Dependability Quality Attributes General Scenarios

Dependability quality attribute availability general and concrete scenarios can be found in section 4.2

- Reliability

Satellite system design specifications (source of stimulus) require the satellite system to handle different kind of failures (stimulus) that could lead to operation disruptions throughout its operation time in the space (environment). Therefore, the system (artifact) should recuperate from disruption (response) to its normal operations (response measure).
Table 32 shows the general scenario for reliability quality attribute.

- Integrity and Confidentiality

The system is controlled remotely from control base.

Elements	Description
Source of Stimulus	Possible Sources of Threats
Stimulus	Various kind of threats.
Simulated artifacts	Satellite Software and Communication Data.
Environment	Normal Condition
Response	System shall secure access and communication data while transmitting.
Response Measure	high level of security to access satellite software and encryption to communication data.

Table 33: Integrity and Confidentiality General Scenario

There will be a large amount of communications from and to the satellite system (Source of Stimulus) during its operation in the space (environment). Communication data is supposed to be secured from different kinds of threats (Stimulus) from unauthorized intruders (Source of Stimulus). Therefore satellite system is expected to have high security at different levels (Response Measure) on its data and programs (Artifact) and be able to prevent these threats (response) from interruption or alternation in its data or programs during its service in the space.
Table 33 shows the general scenario for integrity and confidentiality quality attributes.

- Safety

Risk management of satellite system proposed that the system must be able to handle the fact that during its operation there are a large number of mishaps from control-base staff or failures in the system components (Source of Stimulus). The satellite system is expected to be self-cognizant and shall have control (Response) on the mishaps or the failures (Stimulus) in the system components so it does not result in any catastrophic consequences neither to the satellite system itself or others in the environment or the surrounded environment (Environment). Therefore, the system is expected to be at a high level of safety standards when its operating (Response Measure).
Table 34 shows the general scenario for safety.

- Maintainability

Elements	Description
Source of Stimulus	Satellite control base staff and Satellite system components
Stimulus	Mishaps and Failures.
Simulated artifacts	Satellite System.
Environment	Normal Condition
Response	Self-cognizant and have control when mishaps or failures are encountered.
Response Measure	Achieve high level operation standards.

Table 34: Safety General Scenario

Elements	Description
Source of Stimulus	Satellite control base.
Stimulus	Software and Hardware Upgrades.
Simulated artifacts	Satellite System components.
Environment	Normal Condition
Response	Ability to upgrade remotely (online).
Response Measure	Duration of achieving necessary upgrades shall not exceed a duration of one week.

Table 35: Maintainability General Scenario

The satellite system proposed to operate for a long period of time (Environment). During this long life, the satellite system is most likely required to be upgraded (Stimulus) from control base maintainers (Source of Stimulus). The system (Artifact) shall provide the opportunity to be upgraded remotely (Response) and shall not exceed the duration of maximum one operation week to maintainer to implement upgraded features or functionalities (Response Measure).
Table 35 shows the general scenario for maintainability quality attribute.

A.4 Dependability Quality Attributes Concrete Scenarios

- Reliability
Scenario I - (Maturity)

SMART1-2005 system hardware and software components are required to be cutting-edge technologies. Due to mission (Source of Stimulus) conditions (Stimulus)the system both hardware and software

Elements	Description
Source of Stimulus	Satellite operation condition.
Stimulus	Different conditions and system components failure frequency.
Simulated artifacts	Satellite System components.
Environment	Normal Condition
Response	Minimum rate of failure frequency.
Response Measure	1/100000 Hour as maximum rate of failure frequency.

Table 36: Maturity Concrete Scenario

Elements	Description
Source of Stimulus	System Design Requirements.
Stimulus	System components malfunction.
Simulated artifacts	System hardware and software components.
Environment	Normal Condition
Response	Maintain response time.
Response Measure	Response time shall not exceed 0.01 second per request.

Table 37: Fault Tolerance Concrete Scenario

components (artifact) are required to have a minimum rate of failure frequency (Response) that does not exceed 1/100000 Hour (Response Measure) when it's operating in normal conditions (Environment).
Table 36 shows the concrete scenario for reliability quality attribute sub-characteristics (Maturity).

Scenario II - (Fault Tolerance)
According to design constraints (Source of Stimulus) SMART1-2005 hardware and software components (Artifact) response time shall not exceed 0.01 second per request (Response Measure) in its normal condition (Environment). However, the system is required to maintain its response time (Response) in case a component has a malfunction (Stimulus) during its operation.
Table 36 shows the concrete scenario for reliability quality attribute sub-characteristics (Fault Tolerance).

Scenario III - (Recoverability)
Reliability of satellite components is a major issue due to its operation rules (Environment). The system operation environment or users (Source of Stimulus) might cause a failure (Stimulus) in system components (Artifact). The failed components are expected to recover from the failure mood (Response) within a maximum amount of 10 minutes (Response Measure).
Table 38 shows the Concrete scenario for reliability quality attribute sub-characteristics (Recoverability).

Elements	Description
Source of Stimulus	Operation Requirements.
Stimulus	Failures that might be caused due to satellite operation environment.
Simulated artifacts	System components.
Environment	Normal Condition
Response	Recover by resetting to its previous state.
Response Measure	recovering time shall not exceed 10 minutes.

Table 38: Recoverability Concrete Scenario

- Integrity

SMART1-2005 communicates with the control-base transmitting processed requests. The transfer of results while communicating with control-base is requested to be encrypted. This encryption shall be done at both levels encryption of communication signals and encryption of transferred data (Artifact). Both stratagems are supposed to guarantee that satellite possess high capability (Response Measure) of preventing (Response) infraction interrupting or alternation (Stimulus) to communication data between satellite and control-base during its operating in the space (Environment).
Table 39 shows the concrete scenario for the dependability quality attribute Integrity.

- Confidentiality

The system security policy requires SMART 1-2005 system data and programs to be secure (Source of Stimulus). System proposed to secure the use of its function features and databases (Stimulus). The system (Artifact)shall obtain a high level of capability (Response Measure) in preventing or blocking (Response) any unauthorized access (Stimulus) to the aforementioned resources during its operation in the space (Environment).

Elements	Description
Source of Stimulus	Intruders to the satellite communications.
Stimulus	interruption or alternation.
Simulated artifacts	communication signals and transferred data.
Environment	Normal Condition
Response	High level of encryption.
Response Measure	Prevention of interruption or alternation to communication signals and transmitted data.

Table 39: Integrity Concrete Scenario

Elements	Description
Source of Stimulus	Security policy.
Stimulus	Use of satellite functions and features.
Simulated artifacts	Satellite System.
Environment	Normal Condition
Response	High level of authenticating unauthorized access.
Response Measure	Prevention and reporting abuse or attempt of satellite system features and functions.

Table 40: Confidentiality Concrete Scenario

Table 40 shows the concrete scenario for the dependability quality attribute confidentiality.

- Safety

SMART 1-2005 (Artifact) functionalities are controlled remotely by control-base. According to risk management specifications (Source of Stimulus) the system must be self-cognizant by identifying and preventing (Response) control commands that could result in catastrophic consequences (Stimulus). Moreover, the system shall successfully secure the satellite from this type of control commands (Response Measure) during its operation orbiting the moon or returning to earth (Environment).

Table 41 shows the concrete scenario for the dependability quality attribute safety.

- Maintainability

Elements	Description
Source of Stimulus	Risk Management on satellite operation.
Stimulus	Commands that could result in catastrophic consequences or serious damage.
Simulated artifacts	Satellite System.
Environment	Normal Condition
Response	Self-Cognizant Identification of commands.
Response Measure	Success in preventing and blocking commands that may result in catastrophic consequences and serious damage to satellite system.

Table 41: Safety Concrete Scenario

Elements	Description
Source of Stimulus	Stakeholders and Design Requirements.
Stimulus	Modification to the satellite system by adding additional features.
Simulated artifacts	Satellite System Platform.
Environment	Normal Condition
Response	Designing satellite system platform capable of remotely modifications to satellite system.
Response Measure	successful modification within a time of an operational week and less amount of effort.

Table 42: Expansion Capability Concrete Scenario

Scenario I - (Expansion Capability)

Due to design specification and stakeholder requirements (Source of Stimulus) SMART 1-2005 system components are required be modified having additional features (Stimulus). The system platform (Artifact) shall be designed to have a high capability (Response) to allow control-base developers to add extra features or components with less amount of effort and time estimated by an operational week (Response Measure). Moreover, system modifications will be done remotely (Environment).

Table 42 shows the concrete scenario for the dependability quality attribute maintainability subcharacteristics (Expansion Capability).

Scenario II - (Flexibility)

Elements	Description
Source of Stimulus	Developers.
Stimulus	Remotely (on-line programming) upgrading satellite software system.
Simulated artifacts	Satellite System Software.
Environment	Normal Condition
Response	System is capable to providing an on-line upgrading mode to control base developers.
Response Measure	Accomplishing upgrading within a duration of maximum a week.

Table 43: Flexibility Concrete Scenario

When there are planned or unplanned maintenance for SMART 1-2005 system software (Artifact). Maintenance policy requires the system to provide the opportunity (Response) for system developers (Source of Stimulus) to remotely (Environment) upgrade via online programming (stimulus). The upgrade shall require a minimum amount of developers time estimated as a week of programming by control-base developers (Response Measure).

Table 43 shows the concrete scenario for the dependability quality attribute maintainability subcharacteristics (Flexibility).

B High-level Embedded Systems Quality Attributes Specifications

Functionality: a set of attributes that relate to the existence of a set of functions and their specified properties. These functions satisfy stated or implied needs.

1. *Security*: a set of embedded systems software and hardware attributes that relate to the system ability to prevent different types of unauthorized access to programs or data.
2. *Accuracy*: a set of embedded systems software and hardware attributes that reveal the provision of right or agreed results or effects.
3. *Compliance*: attributes of embedded systems software or hardware that make the system adhere to applications related to the stated technical and business rules. (expansion capability in terms of system functionality).

Efficiency: a set of attributes of embedded systems hardware and software that relate to the relationship between the level of performance of the system and the amount of resources used, under stated conditions.

1. *Resource Behavior:* Attributes of the system hardware or software that relate to the amount of resources used and the duration of such a use in performing its function. (Code Size, Memory Usage, Peripherals, System Resources, Physical Size, and Power Consumption)
2. *Time Behavior:* Attributes of ES hardware or software that relate to response and processing times and on throughput rates in performing its functions. (CPU -Time used and Response Time).

Maintainability: a set of embedded systems software or hardware attributes that relate to the effort needed to make specified modifications.

Maintainability quality attribute in the context of embedded systems has the following sub-characteristics:

1. *Expansion Capability:* Embedded system software or hardware attributes that relate to the level of system capacity and effort needed to add functions or hardware components, under stated conditions. (in terms of embedded systems maintainability)
2. *Flexibility:* a set of embedded system software or hardware that relate to the capability of the system to be modified without breaking.

Although there are only few descriptions dedicated to embedded systems reliability quality attribute, ISO 9126 provide three quality sub-characteristics which can be interpreted for embedded systems reliability as the following:

Reliability: a set of embedded systems software or hardware attributes that relate to the capability of the system to maintain its level of performance under stated conditions for a stated period of time "Duration".

1. *Maturity:* Attributes of embedded systems that relate to the frequency of failure in both hardware and software components.
2. *Fault Tolerance*: Attributes of embedded system software and hardware that relate to their ability to maintain a specified level of performance in case of components failure or infraction of its specification.
3. *Recoverability:* Attributes of embedded system software and hardware that relate to their capability to re-establish its level of performance and recover the parts which are directly affected.

Portability: a set of attributes that relate to the ability of the embedded system software and hardware to be transferred from one environment to another.

1. *Ease of Integration:* a set of embedded system hardware and software attributes relate to the opportunity for its adaption to different specified environments and being adhered to standards related to portability.

2. *Compatibility:* Attributes of the embedded system hardware and software that relate to effort needed to install the system in a specified environment or using it in the place of specified other systems in the environment of that system.

Usability: A set of embedded systems software and hardware attributes that relate to the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

The ISO 9126 provides three sub-characteristics that support the quality attribute of usability. The translation of theses sub-characteristics are done as the following:

1. *Understandability:* Attributes of embedded systems software and hardware that relate to the user's/ operator's effort to recognize the logical concept and its applicability.

2. *Learn-ability:* attributes that relate to the effort needed from the user to learn the system applications.

3. *Operability:* a set of embedded systems attributes that relate to the effort required from the user for operations and operation control.

C Research Surveys

C.1 SURVEY I: Embedded Systems Quality Attributes Translation Survey

Embedded systems quality attributes found in T. Sherman Study according to software quality models such as McCall, Boehm, and ISO 9126 are considered quality criteria or quality factors characteristics [1]. In order to be able to apply quality attributes relationship reasoning these quality attributes are required to be elevated to a higher level of specifications and details. In this study, we have selected the ISO 9126 quality model.
Candidate participants for this research survey were carefully selected master students in the field of software engineering. The methodology for sampling in this research is stratified sampling method [37]. The population consists of 15 master students who have completed (60 ECTS) of their master program. The research survey is distributed via electronic mails. The process duration of answering research survey and reply researcher with results is estimated to one working week. Data collection and processing is conducted by matching participants survey answers for the appropriate representation of embedded systems quality sub-characteristics/criteria to a higher level embedded systems quality attribute.

C.1.1 Supplementary Data and Survey Question

Participants are supported with quality attributes specifications according to the ISO 9126 quality model. These specifications are extracted from [24].

The survey main question is constructed to be a straightforward question. The question is in the form of a table were participants are to fill in the blanks as the representation of embedded systems quality attributes in a high level using ISO 9126 quality model. Table 44 represents Research question. Table 3 depicts answered question and the rounded level of agreement of participants answers.

Research Survey Main Question :

- Using the supplementary data found in section I of this survey please fill in the table "Embedded Systems Quality Attribute Translation Using ISO 9126 " the suitable representative embedded systems high-level quality attribute.

C.2 SURVEY II: Dependability Quality Attributes Candidate Architectural Tactics Impact Evaluation

As stated in the research paper, dependability quality attributes candidate architectural tactics are based on experience. Therefore, the survey was set to be distributed among carefully selected participants. Participants in this survey are from different software development backgrounds such as academia, software designers, embedded systems software programmers, software engineers and security engineers. The survey design focuses on two elements: supplementary data and the research question. Research survey was pretested by the author before executed. The author have established different prototypes of this survey to guarantee a systematic and strait forward quantitative data collection and processing. The prototypes were executed on four software engineering master students and process their feedbacks and suggestions for developing a better quality survey. Moreover, the author examined research survey technical issues according to research survey's pretest suggestions found in [41]. The survey pretest was focused on survey enclosed supplementary data, survey question, survey answer mechanism explanation, survey participants sampling, and survey data collection. However, the research survey was divided into four main parts according to dependability quality attributes candidate architectural tactics and the participants background:

- Availability and Reliability Survey : This survey was distributed among participants of embedded

Table 44: Research Survey Question in the Form of Table

Embedded systems Quality Attributes Found in [18]	Quality Attribute Type	Translation to ISO 9126
Code size	Software Quality	
Memory usage	Software Quality	
CPU Time Used	Software Quality	
Accuracy	Software & Hardware Quality	
Compatibility	Software & Hardware Quality	
Complexity, CPU, System, Peripherals etc.	Software & Hardware Quality	
Cost/Schedule	Software & Hardware Quality	
Ease of Integration	Software & Hardware Quality	
Ease of Use/Usability	Software & Hardware Quality	
Expansion Capability	Software & Hardware Quality	
Functionality	Software & Hardware Quality	
System Resources Usage	Software & Hardware Quality	
Versatility/ Flexibility	Software & Hardware Quality	
Performance	Software & Hardware Quality	
Availability	Software & Hardware Quality	
Reliability	Software & Hardware Quality	
Safety	Software & Hardware Quality	
Security	Software & Hardware Quality	
Maintenance	Software & Hardware Quality	
Durability	Software & Hardware Quality	
Physical Size	Hardware Quality	
Power Consumption	Hardware Quality	

system development, academic and software design backgrounds.

- Integrity and Security Survey: The survey was distributed among participants with security engineering backgrounds.

- Safety Survey: This survey was dedicated to participants of software engineering background.

- Maintainability: This survey was distributed among participants of software engineering background.

Additionally, the investigator has performed his own evaluation answering the aforementioned surveys. Participants were given sufficient time to answer research survey and provide research with their feedback. The process of answering the research survey is estimated to be 1 to 2 weeks.

C.2.1 Survey's Supplementary Data

In each one of the research surveys, supplementary data consists of three parts:

- Dependability and Embedded Systems Quality Attributes General and Concrete Scenarios: See appendix A.
- Embedded Systems quality attributes specifications: See Appendix B.
- Architectural tactics specifications: we have provided participants with architectural tactics specifications according to [5][29].

C.2.2 Survey Question

The survey question is an iterative question to allow participants to evaluate each candidate architectural tactic impact on dependability and embedded systems quality attributes separately. We have provided the participants with an example to show them the desired mechanism to answer survey question in addition to the defined scale to evaluate the level of impact of a proposed tactic. Table 45 and 46

shows an example of research question for a given architectural tactic X to achieve availability and reliability dependability quality attributes.

Research Survey Main Question :

Note: Please, Kindly use necessary information in section I (Supplementary Data) to answer the question.
What is the impact of using Tactic X on embedded systems and dependability quality attributes , use the pre-defined scale to give a measured evaluation for the level of impact ?

Please justify your evaluation result of the impact on quality attributes in Table 46.

Table 45: Availability and Relaibility Architectural Tactic (X)Impact Evaluation

Quality Attributes	Functionality	Efficiency	Usability	Portability	Integrity	Confidentiality	Safety	Maintainability
Impact Nature								
Impact Level								

Table 46: Availability and Reliability Architectural Tactic (X) Impact Evaluation Justification

Quality Attributes	**Please justify the impact evaluation on quality attribute Here !**
Functionality	
Efficiency	
Usability	
Portability	
Integrity	
Confidentiality	
Safety	
Maintainability	